Universidad Carlos III de Madrid

Aerospace Department

Bachelor Thesis

Design and implementation of an EyasSat command and control center

Author:
Daniel Expósito Jiménez

Supervisor:
Dr. Mario Merino Martinez

September 27, 2015

Acknowledgement

I wish to express my gratitude to the supervisor of the project, Dr. Mario Merino, for being patient enough to teach me how to improve myself in that long journey. I would also like to say "thank you" to Manuel Sanjurjo and Filippo Cichocki, who lent me some of their time so that I could give my best.

In a more personal way, I would like to express my gratefulness to all those that have been next to me in these four years. My classmates, for becoming my new family where you can always find your place, specially Andrea Grande y Tamara Casillas. My friends, for knowing how to handle with my perseverance and my desire for self-improvement. Finally, in a more special way, I would like to thank my family, for showing me the way to become who I am.

Contents

List of Figures

Chapter 1

Introduction

1.1 Space systems. Spacecraft subsystems

A space system is the combination of the entities and elements that make possible the development of some predefined operations in the expanse beyond Earth's atmosphere. The main classification that can be done is regarding its user communities, since they can be classified in *Civil*, *Commercial* or *Military* systems. Concerning the physical structure, there is a very wide variety of spacecraft systems, however its architecture is typically composed by three elements: *Space segment*, *Launch segment* and *Ground Station segment*. Satellites are contained in the space segment, and they carry the payload and the required infrastructure to operate it and to communicate with the Ground segment. Launch segment or Launcher system is in charge of allocating the satellites in their proper orbits and finally the Ground segment or the Supporting Ground Control System will take care of the spacecraft, its environment, the fulfillment of its mission and its proper operation. Engineers should confront the task of making all the segments of the systems fulfill the requirements of the mission in an optimum way, achieving a complete integration of the different units.

Regarding the Space segment classification, it can be split in two categories, *payload* and *bus*. The *payload* is the motivation of the mission itself. Within the *bus* field of the classification all the subsystems needed to accomplish the requirements of the mission are contained. A common classification can be seen in Figure 1.1.

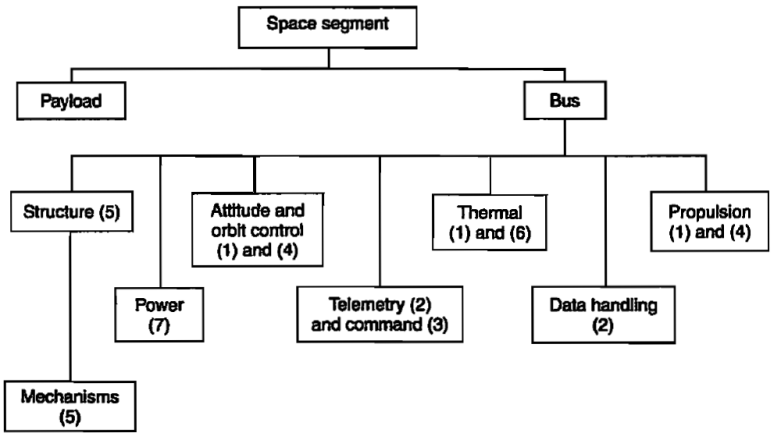


Figure 1.1: Subsystems' classification

Clearly, depending on the mission, the subsystems will differ slightly but all of them need to be present and the engineer should know quite well how they work and how they are integrated. In future chapters the reader could know which subsystems are involved in this project and how they work. For greater detail see also [1].

1.2 The EyasSat simulator

The Aerospace Engineering department of "Universidad Carlos III de Madrid" is trying to raise a special interest among their students, therefore, it has acquired an EyasSat simulator in order to impart lectures about space systems in some courses of the "Master in Aeronautical Engineering".

The EyasSat is a small and versatile satellite just created for educational purposes, which allows students to practice and learn about space systems and integration. The name " EyasSat " comes from falconry. " Eyas " means " baby falcon " or " fledgling bird." The falcon is the mascot of the U.S. Air Force Academy, where the concept of EyasSat was born. The first EyasSat was co-developed between the U.S. Air Force Academy and Colorado Satellite Services and it is built and owned by Eyassat LLC.

Figure 1.2: EyasSat's structure

EyasSat is composed by some boards installed one over each other, joined all of them by a central bus. This combination is covered by a squared structure that brings some of the subsystems and also protects the inner part where there is a wheel mounted, that allows the rotation around its vertical axis. The final assembly can be seen below in Figure 1.2. In the following chapters it will be entirely described and defined.

1.3 Objectives of this work

1. Learn the functions and working principles of the systems and subsystems of a spacecraft.
2. Enhance the knowledge about "Project Engineering" of the student.
3. Become familiar with the EyasSat, its parts and its performance.
4. Learn how to command EyasSat manually and from its own GUI.
5. Learn how to perform serial communication (RS-232) with the EyasSat.
6. Set up communication with EyasSat developers.
7. Learn to program and use acquisition data software such as LabView.
8. Use the acquired knowledge in LabView to program a GUI that could replace the EyasSat's own GUI. The new GUI should allow the user to interact with the

satellite, both through serial connection to the ground station and with radio link through the communication module. The interface should be capable of: knowing the state of the satellite at any instant of time, commanding any desired change in all these states of the spacecraft and finally communicating with Matlab scripts developed to apply autonomous control laws.

9. Write the Matlab scripts that are inserted in the GUI. These scripts should embrace the control laws that, from the reading of the state vector of the satellite, will define the necessary control command to the LabView interface. The scripts will be embedded in the interface and called from it upon user's request.

10. Know and emphasize the real limitations of the EyasSat satellite.

11. Test and assess the performance of the control laws embedded in the new GUI.

12. Learn to create a formal engineering report and to generate technical documentation using LaTeX.

1.4 Contents of this report

The report begins with an introduction about what is space systems, what are they composed of and how can they be classified. Besides, it briefly introduces the EyasSat system and the main motivations of the work.

In chapter 2 the EyasSat is thoroughly explained. The reader can find an explanation about how its subsystems work and what are their functions, how the satellite communicates with the environment and with its ground station, how it reports information and how it could be commanded. Furthermore, its limitations and some efforts to overpass them will be commented. Since the main subsystem of the EyasSat is the "ADCS" one, the following contents are devoted to explain its study and characterization. This information is encompassed in chapter 3.

Once the system and their subsystems are fully understood, chapter 4 and chapter 5 are dedicated to the comprehension of the behavior of the EyasSat and also to develop the way of modifying that behavior, following user's intentions.

As soon as the software is fully coded, the following actions are based on testing how the software works and how the satellite response their commanding actions. These different trials are comprised in chapter 6.

Finally, the work ends with the conclusions extracted from the process and the experience obtained through it, together with some appendix that complements the information presented in the work.

Chapter 2

Knowing the EyasSat: subsystems, modules capabilities and limitations.

2.1 Global view of the EyasSat

The EyasSat system is developed to simulate a real spacecraft, allowing the students to learn the key aspects of space systems design and get involved in space operations. For that purpose it is composed of the same subsystems as a real spacecraft. The EyasSat is built by different boards, each of them will add a different subsystem. Then, the total stack of boards will be covered by a external structure that adds some subsystems too. The main subsystems will be explained in greater detail later, they are:

- Structural Subsystem
- Electrical Power Subsystem(EPS)
- Command and Data Handling Subsystem(C&DH)
- Communications Subsystem(COMM)
- Attitude, determination and control Subsystem (ADCS)
- Thermal Subsystem
- Additional subsystems

EyasSat has the capability of adding some extra functions just by joining other boards such as the ones presented. Since they can be installed independently, there are some umbilical cords in the system that let the user analyze the behavior of each singular subsystem through serial connection. This can be done just by using a USB port that connects the board to the ground station.

In order to use the capabilities that EyasSat offers, the satellite has its own GUI developed by the company. This GUI consists of 6 different tabs, each of them for a specific subsystem, below the reader could see in Figure 2.1 the main view of the GUI. Actually, one of the tabs, the so-called "Text only interface" is the only one that has been used during that work, since it allows the communication with the satellite.

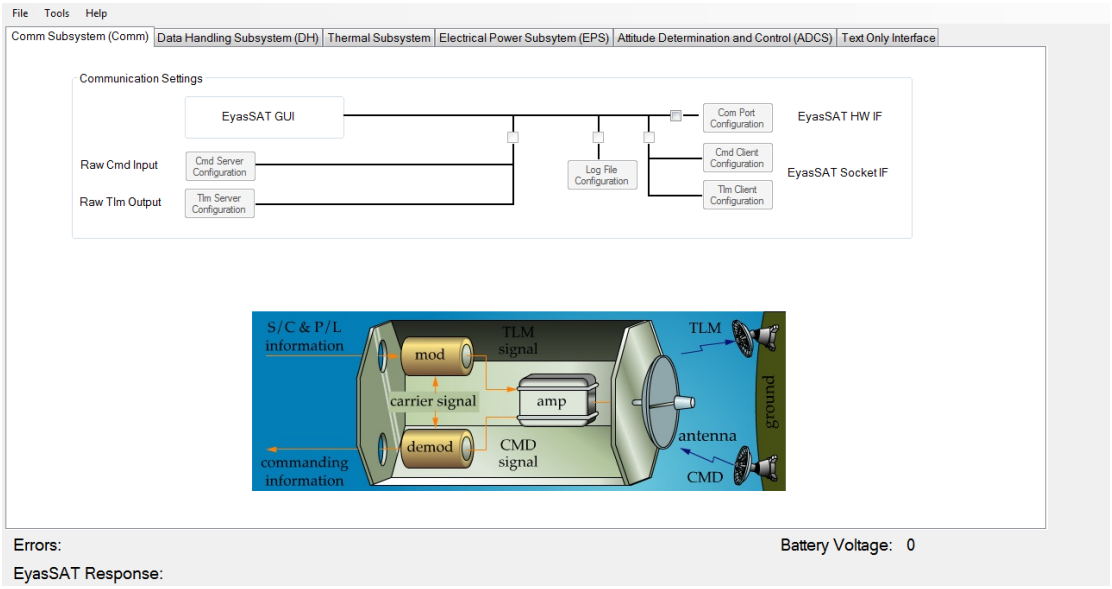


Figure 2.1: EyasSat's own GUI

2.2 Subsystems of the EyasSat

2.2.1 Structure & mechanisms

The primary structure of the satellite fulfills two requirements:

- It allocates and support all the subsystems of the spacecraft, as it is in a stand-alone configuration.
- It protects the subsystem and the payload from the hazards of space and the launch environment.

The prism structure has a squared base whose side lengths 18.5 cm and the prism's height is 20.3 cm. Within that box, all the subsystems can operate safely. Its final integration of the structure is shown in Figure 2.2

Figure 2.2: Complete assembly

2.2.2 Power Subsystem

Power board subsystem is devoted to provide the required electric power to the whole payload to feed other subsystems, as it would be in a real spacecraft, where the only source of energy would be the solar radiation and the batteries. Furthermore, since the satellite is purely educational, the power subsystem is complemented with some umbilical cords. They can power the satellite, load the batteries and also feed the subsystems for further and individual studies.

This subsystem is composed by three elements that provide voltage in different ways. The main element is the Electric Power Board, which is the bottom board of the assembly. Within this board the user can find the main bus, used to create the communication between boards, three different pins used to connect the board to the thermal panel, to the solar array and to the sun sensor. The board also possesses a GSE or flight plug, where the arm plug is connected when the satellite is ready to operate and flight autonomously. Additionally, user can find some LED lights, that are used to assess the connection of the different elements to the boards, a data port with the label " UMB " meaning umbilical, where the connection cable that joins the board to the computer is engaged. There are also a SEP switch and some test points that are used to perform some electrical measurements to define properly the electrical properties of the board. Finally, the last element found is the circuitry, that makes possible all the related functions. Below the reader can find an illustration of the system in Figure 2.3.

There is other way to provide voltages to the system, which is using the solar array that is located in one of the external walls of the satellite. This array is formed by 6 panels, located in a matricial distribution of two arrows and three columns. In order to understand how these cells work, reader could find more information about solar cell physics in A.

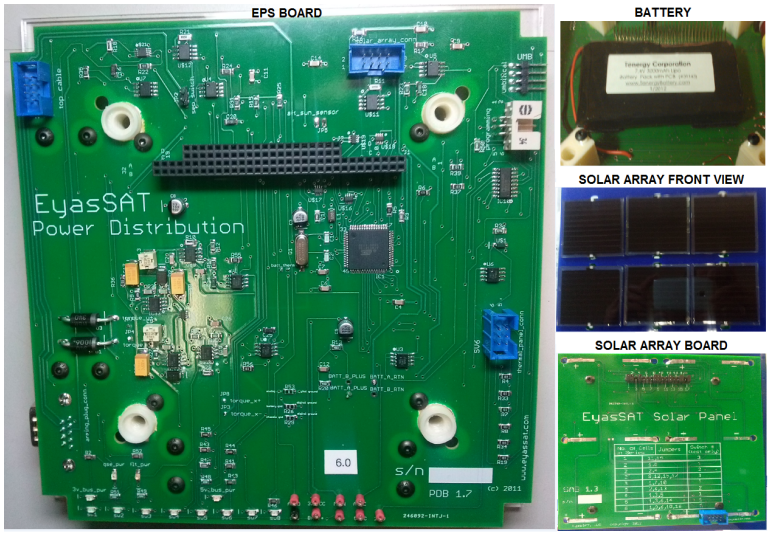


Figure 2.3: EPS subsystem

The third and last element that provides power to the satellite are the batteries, that are located at the bottom of the satellite. They can be loaded through the GSE plug in the ground station of the EyasSat either through the solar panels that are installed

on the system. They can be visualize Figure 2.3.

2.2.3 Command and Data Handling

The C& DH board is installed to provide the satellite with an internal unit able to read the on-board sensors, collect all the measurements and writing them into the telemetry lines so that the user can read them easily. Besides, it allows the user to be in contact with all the subsystems of the device, creating the adequate channel through which the different boards are read and commanded. In a real spacecraft this subsystem is used to acquire and to post-process the data that should be obtained during the whole mission, concerning the EyasSat is just to recognize the environment.

That board is compound of the different elements that can be appreciated in the Figure 2.4.

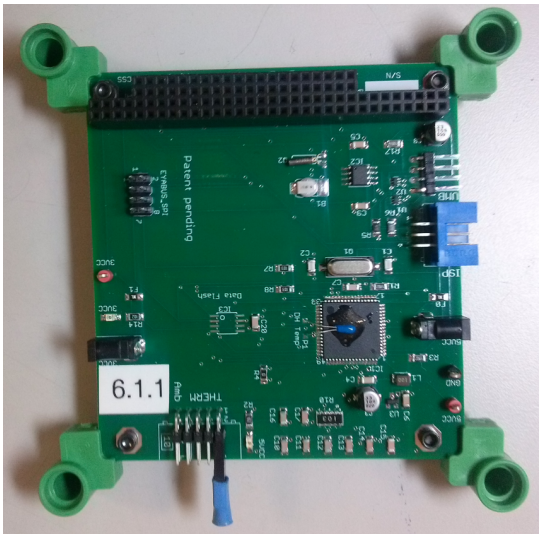


Figure 2.4: C& DH Module

Within the board the user can find: an input power port in case the user wants to communicate and test only the C&DH board, a data port through which it is connected to any computer in order to transmit and receive information. There is one real time clock, which allows the CPU to insert next to each message the date and the hour, so that the messages that are stored and produced can be classified in a proper way. There are two different transistor installed in the board, there is one of them devoted to read a difference in voltage and convert it into a measurement of the ambient temperature. In parallel, the other transistor is installed with another component, the processor. Having that thermistor installed next to the CPU allows the user to check continuously its health. The processor is in charge of handling the whole bunch of information and its transmission. There are some onboard sensors distributed through the whole system that send their information directly to the Data Handling CPU through the main bus that connects all the boards together. This is how the CPU constructs the telemetry lines the user will see.

2.2.4 Communications

The communication subsystem basically what it does is transmitting and receiving some electromagnetic signals that are post processed and interpreted in the main CPU of the receiving device. This subsystem works similar to a real spacecraft's one, creating the link of communication between the satellite and the ground station. The main board of the subsystem can be appreciated below in the Figure 2.5.

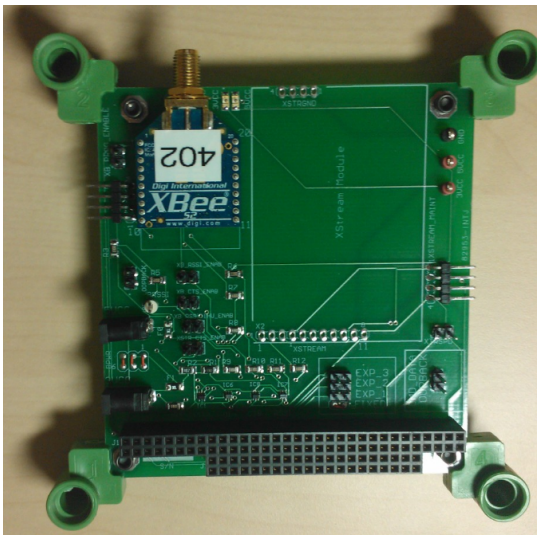


Figure 2.5: Communication board

The major component of that board is the central bus that joins all the boards and allows their intercommunication. Apart from that way of transmitting information, there is also a data port that makes possible the connection between the single board and any other computer. Since the single connection of the board is allowed, clearly there should be some power ports in order to feed the subsystem. In concordance with this, some voltage pins can be seen on the board, so that user or students can perform some analysis or calibration of the board's power. Finally, the only element missing some explanation is the submodule labeled as XBee. XBee is a company that manufactures electronic products devoted to communications. Actually that module allows the satellite to flight autonomously without losing the communication and transmission of information with it. The radio has a data rate of 250 kbps and uses a frequency of 2.4 GHz. However, during an autonomous flight, the satellite should be fed through the internal batteries that it carries with it, so that it must have some protective system to prevent fatalities. This is why the board has a loop-back jumper.

Figure 2.6: Radio ground support

During the standalone missions that can be assigned to the EyasSat when the propulsive module is installed into it, the whole system will be covered with the external housing, and therefore the antenna should be located in another place. Actually,

the antenna will be installed in the top panel of the satellite. For that purpose, there are some pin buses that are connected in the structure, allowing the flow of information through the whole satellite. The standalone configuration of the satellite is shown above in Figure 2.2.

There is one more device that is appreciable in the following figure, that is the ground module of the radio that is connected to the ground computer. It is shown here in Figure 2.6.

2.2.5 ADCS

This is one of the most complex subsystems of the satellite and the central one in the work, this is why Chapter 4 is devoted to it. It determines the attitude of the spacecraft using onboard sensors. In a real spacecraft that subsystem is devoted to modify the orientation of the satellite so that the proper attitude is selected depending on the environment or on the solar radiation, so that the satellite could charge its batteries. The actuator should be characterized with a well-defined PID controller so that the ground station could determine and modify the position of the spacecraft quickly in a stable way.

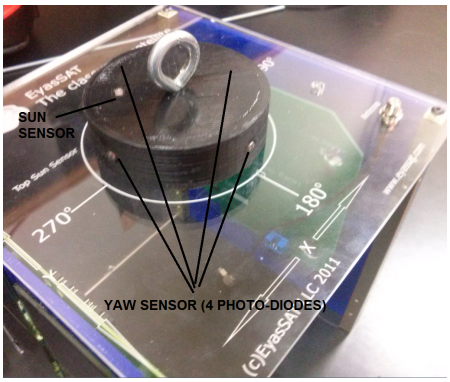


Figure 2.7: External ADCS sensors

Within the EyasSat subsystem there are two photosensors located at the top and bottom of the satellite and there are other four that are located in the top panel just to compute the yaw angle with respect to the light source. They are capable of measuring the intensity of the light, which is used by the EyasSat to compute the attitude of the satellite in the three axis with respect to the source of light. They can be seen in Figure 2.7.

Moreover, the board is complemented with the reaction wheel. It allows the satellite to modify its orientation using its spin rate and using some magnetometers that are installed to help the wheel to relax some loads. In order to ensure the measurements are taken properly and they are accurate enough, the spacecraft's board also has a gyroscope and an accelerometer.

Additionally, this board has the central bus which is needed to perform the communication between boards, and also its own CPU to take, translate and transmit the required information. The reader can find a graphical representation of the system below in Figure 2.8.

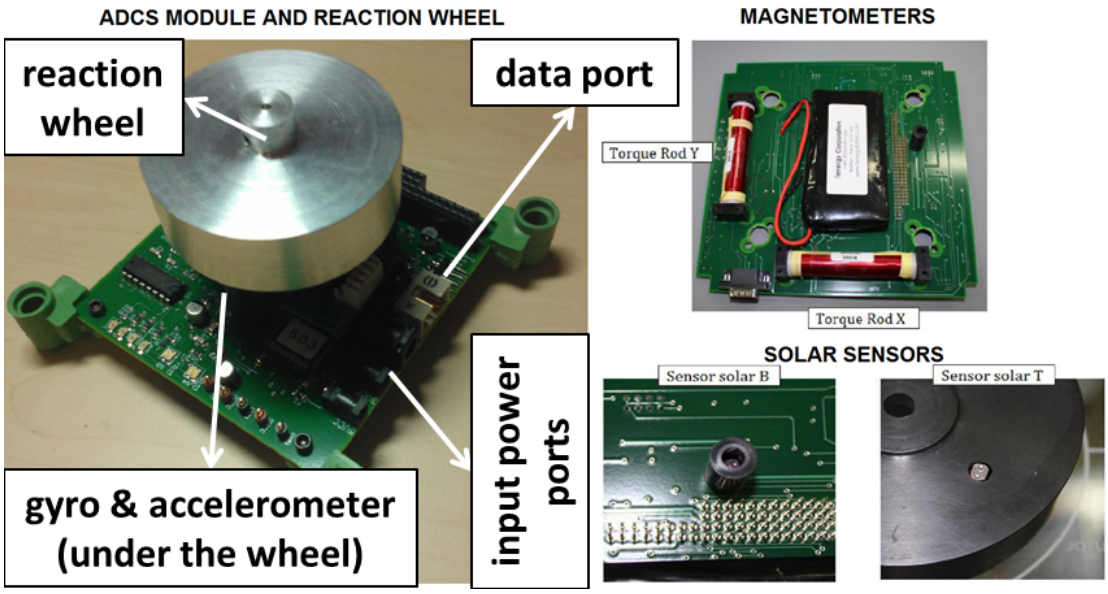


Figure 2.8: ADCS board

2.2.6 Thermal control

During the realization of this work the thermal control subsystem were not used. However, it should be known and integrated. It is composed of two flat surfaces, equal but different in colors. They are black and white. The aim of these small panels is measuring how the different absorptivities and emissivities of different surfaces have an influence on the steady-state temperature under solar light illumination. Next to the solar panels there are two pipes of different material that allows the user to study how thermal energy is affected and transmitted through the materials. One pipe is full-copper, the other is a heat pipe. It is a hollow pipe with a wick and an alcohol that is in equilibrium with its vapour phase; you can read more about it in the books. This subsystem can be appreciated in the Figure 2.9. Additionally, there are some thermal sensors in the spacecraft, so that there is complete knowledge about its temperature and therefore its health all the time.

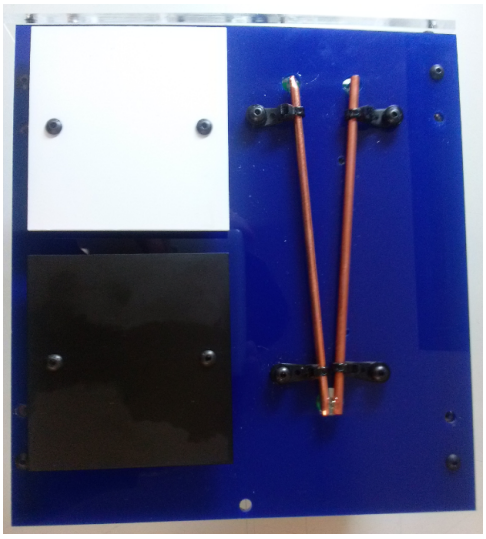


Figure 2.9: Thermal subsystem.

2.3 EyasSat telemetry and command codes

2.3.1 Communication with the EyasSat

In order to communicate with the EyasSat, the first attempt should be performed using its own GUI that was shown in Figure 5.1. The communication channel of the EyasSat is bidirectional, i.e, it can receive and transmit information, however it cannot do it simultaneously, therefore it is a half-duplex system of communication.

The data that is being transmitted by the EyasSat is the telemetry data, and it refers to the conditions of any state variable that is controlled by the satellite sensors.

On the other hand, the information that is updated into the satellite are the telecommands that are written by the user. They are already defined since the EyasSat has its own language and they will be presented below. As explained in Chapter 2 the telecommands can be transmitted through the serial RS-232 or using the XBee module developed for that purpose.

2.3.2 Description of the telemetry lines

When the communication is established with the EyasSat as a whole, and even with each of its entities, i.e, its boards, the system starts to send some information that it takes autonomously from its environment. It sends a bunch of information that in a first sight could seems muddled, therefore, during the following lines, this message will be shown and decoded so that it could be easily understood.

```
ES0 03:36:57 I_TelemDelay=10 CmdTimeOut=3 Pwr=1 ADACS=1 Exp1=0 Exp2=0
Exp3=0 ES0 03:36:57 T DH=20.9 Sp=19.9 Exp=-4.7 Amb=19.4 ES0 03:36:57
P Sep=1 V_Batt=7.36 I_Batt=-202.00 V_SA=7.39 I_SA=0.00 I_MB=215.00
V_5v=4.93 I_5v=88.44 V_3v=3.27 I_3v=98.00 S=20 T_Batt=18.14 T_TopA=17.61
T_TopB=17.79 T_Base=17.44 T_Wht=17.79 T_Blk=17.8 ES0 03:36:58 A X=0 Y=0
sunT=756 sunB=1 p1=616 p2=717 p3=726 p4=687 ya=164.7 sa=0.0 rate4x=0.0
rate1x=0.1 rps_out=0.0 rps_cmd=0.0 PWM_out=0.0 alg=0 P=1.0 I=0.1 D=0.1
deltaT=1 deadband=10.0 slope=4.0 offset=20.0 extra=10.0
```

The complete message is always made of four lines, one for each board that is installed in the spacecraft. Each line of the message always starts with the code ESX, where X is the number of the channel that the S/C is using to transmit. Besides there is always a letter that determines which board it refers to. The letters are I, T, P and A, referring to the telemetry coming from the C& DH, the thermistor input bank on the C& DH, the power signal and the ADCS outputs respectively.

- The first line involves the following elements:

Parameter	Description
TelemDelay	Number of seconds between a telemetry message and the following one.
CmdTimeOut	Number of seconds DH board waits for a command.
Pwr	Indicates if power board is attached.
ADACS	Indicates if ADACS board attached.
Exp1,2,3	Indicates if Experiment 1, Experiment 2 or Experiment 3 board is attached.

Table 2.1: Description of parameters for the telemetry's I-line.

- The "T" line provides the temperature in celsius of different elements, they are:

Parameter	Description
DH	EyasSat DH Module.
Sp	EyasSat Solar Array.
Exp	Experiment-this is the thermistor plugged into the connector on the side of the board.
Amb	Ambient temperature.

Table 2.2: Description of parameters for the telemetry's T-line.

- The "P" line refers to:

Parameter	Description
Sep	Separation switch status (0 or 1)
V_{Batt}	Battery module voltage (V)
I_{Batt}	Battery module current(mA)+ is charge and - is discharge
V_{SA}	Solar panel voltage (V)
I_{SA}	Solar panel current (mA)
I_{MB}	Main bus current.
V_{5V}	5V converter output.
I_{5V}	5V EPS line current. (mA, all lines combined)
V_{3V}	3V converter output.
I_{3V}	3V EPS line current. (mA, all lines combined)
S	Reserved parameter
T_{Batt}	Battery Temperature.
T_{topA}	Top temperature of copper rod.
T_{botB}	Top temperature on heat pipe.
T_{BASE}	Temperature of the base of the copper rods.
T_{white}	Temperature of the white solar panel
T_{Blk}	Temperature of the black solar panel

Table 2.3: Description of parameters for the telemetry's P-line.

- Finally the line A refers to the following variables:

Parameter	Description
X and Y	X or Y torque rod status (0, +1 or -1).
sunT	Top sun sensor (no units).
sunB	Bottom sun sensor (no units).
p1, p2, p3, p4	Yaw sensor measurements.
ya	Yaw angle derived from p1,p2,p3,p4.
sa	the offset angle set.
rate4x	4x output of the gyro.
rate1x	1x output of the gyro.
rps_{out}	RPS that EyasSat measures its wheel is moving.
rps_{cmd}	RPS that EyasSat tries to provide to the wheel.
PWM_{out}	PWM commanded to the wheel.
alg	Mode of controlling the wheel (0,1,2).
P	P gain for the closed loop control algorithm.
I	I gain for the closed loop control algorithm.
D	D gain for the closed loop control algorithm.
deltaT	integration time in PID control loop.
deadband	deadband for sun point algorithm.
slope	slope for converting RPS to PWM_{out}.
offset	offset from sun point.
extra	kick value to overcome wheel friction.

Table 2.4: Description of parameters for the telemetry's A-line.

2.3.3 Description of available commands

During the accomplishing of this project, when handling with the EyasSat and its boards, some useful instructions must be given to it. These commands are synthesized into some combinations of different letters that the satellite can recognize. The main group is formed by two letters and some digits. The first character is devoted to recognize which is the board where it is sent, for instance, the commands that start with the letter "a" are destined to the ADCS board. The second character defines the action that such board should carry on. Finally the digits are engaged to the definition of this specific command if it involves some number, such as the wheel's speed definition, or to select between different options that EyasSat brings, for instance the kind of controller that commands the wheel. Below the reader can find a table where all the available combination of commands are explained:

Command	Description
ch	Set hours
cm	Set minutes
cs	Set seconds
eX	Activates experiment X, where X can be 1,2 or 3 respectively
is	Call sign number (1 to 10)
it	Set command TimeOut(seconds) (3 to 10)
ik	Command retries (0 to 10)
id	Telemetry delay (1 to 60)
T	SPI request for tlm
t	UART0 request for tlm
R	Reset from SPI
r	Reset from UART0
S/s	Tlm scaling off or on (0 or 1)
1-8	Turn on/off power swx 1 through 8 (0 or 1)
%	Coefficient programming (0=no, 1=yes)
m	Intercept in $Vout = m + n * ADC_value$ of V_5v
n	Slope in $Vout = m + n * ADC_{value}$ of V_{5v}
x	Intercept in $temp = x + y * ADC_{value}$ of thermistor
y	Slope in $Vout = x + y * ADC_{value}$ of thermistor
!	Resets m,n, x, and y to the default values
m	Telemetry mode (0, 1, 2)
x	X torquer is off, + or - (0, 1, 2)
y	Y torquer is off, + or - (0, 1, 2)
w	This sets 'W'heel RPS value. (-50 to 50)
h	Hysterisis setting or deadband on the RPS commanded value(1 to 10)
c	Closed loop control algorithm(none,PID,"suntrack" control)(0,1,2)
o	Sets the offset angle for suntrack mode (i.e. angle from the sun)(+-179)
p	The value for proportional control in PID loop for RPS control
i	The value for Integral control in PID loop for RPS control
d	The value for differential control in PID loop for RPS control
l	Loop control deltaT time constant for PID (seconds)
b	Deadband for sunpoint control, typically 1 to 5 degrees
a	The is the intercept in the equations $PWM_{out} = a + g * (RPS_{cmd})$
g	The is the gain in the equations $PWM_{out} = a + g * (RPS_{cmd})$
e	Extra boost in PWM used to control motion in sunpoint.

Table 2.5: Description of the available commands for EyasSat.

2.4 Limitations of the system

Unfortunately, the system has some limitations that influences the response obtained from it and the way of approaching its control. They can be found in different subsystems of the satellite and they will be commented below. The first limitation found in the system concerns the telemetry lines and the information reported on it. Readings are barely accurate, not because of the measurements, which are precise enough as will be demonstrated later, but because of the configuration of the inner software. When the

satellite reports each telemetry line and the software reads it from its buffer, it should find a termination character after the whole message, which clearly defines where the telemetry lines finish. However, after spending some time in different trials to guess which termination character is the one used by the system, there were not a clear solution that could solve the problem. Therefore, some requirements must be fulfilled by the message before being read, just to ensure the proper reading action. Concerning the communications of the satellite, reader should know the channel is bidirectional but half-duplex, therefore, whenever the " ground station " is reading from the satellite, no commands can be given to it, implying some delay in communications.

Furthermore, there is another limitation that becomes problematic when the self-controlling functions are applied. The reaction wheel is displaced from its inertial axis, resulting in some frictional forces appearing due to the contact between the wheel and the support and also accompanied with some displeasing noise. Additionally, the commanding actions that are sent to the satellite in order to move the wheel cannot be followed properly. This is because the satellite commands the wheel in terms of " Pulse Width Modulation ", i.e., PWM. This is a modulation technique, that will be explained later on in Chapter 3.

There is another limitation concerning the environment in which the satellite is embedded. Since it does not have a propulsive module, it cannot move freely and therefore to simulate the external conditions in which it could be, it is wired from the top panel. However, the wire exerts some angular momentum distortion in the system due to its stiffness. Therefore, the effect of the wire should be computed and included in the PID analysis to minimize the error that could arise from it.

2.5 Communication with developing company

In order to solve some of the limitations found in the system, communication with the developing company were established. Basically, the main aim of this was the acquisition of some information that could be relevant during the work but they did not provide it when the system was bought. The responsible of the support within the company was called Genah Burditt, she was the person that answered every question made about the EyasSat.

First, they were asked about the physical properties of the reaction wheel, since there are different models depending on which version of EyasSat was bought. They provided information about the material, the dimensions, the inertia properties, etc... however there were some information missed, for instance the maximum torque that could be provided by the reaction wheel.

Although their answers were not too pleasant, their help was needed again when referring the termination character of the telemetry lines. Geenah was asked about it, to see if there were a fixed termination character that the satellite sends, so that some constraints on the telemetry chain could be suppressed and therefore the final "C&CC " would be speed up. Unfortunately, the company answered there is not a clear termination character used by the EyasSat, whether it uses them randomly.

Every time she answers the question, she get the chance to offer new products to buy to her company, without providing a clear answer to the real problems that concerned, therefore the communications were decided to be stopped.

Chapter 3

Characterization of the ADCS subsystem

3.1 Assessment of the ADCS photosensors

The assessment of the sensors is a key step when knowing the capabilities of the spacecraft. A properly-working sensor can make the difference between performing properly a mission or not. A controlling action that should drive the satellite towards the adequate position, could put the satellite in a wrong orientation and let the panels without the required solar radiation, leading to global shutdown.

The first step was checking how the sensors work and calibrating them. As explained, there are six sensors, four of them in the top part of the satellite in order to measure the orientation of the light i.e, the yaw angle, one at top and one at bottom, to calculate the inclination with respect to the source of light. That computation is performed in the internal CPU of the EyasSat but it is also done in the simulators that are explained later in Chapter 4. The process is the following, there are four measurements coming from the photosensors located in the top part of the satellite, going from 0 to 1000. That measurements indicates the intensity of the source of light that is irradiating them. Then, the CPU only takes the greater two, knowing in that way, in which quadrant is the source of light. After that, those values can be translated into the sine and cosine of the angle using a constant value that is predefined by the system. Finally from the sine and the cosine, since the quadrant is known, the angle is fully defined.

EyasSat has its own variable to define the yaw angle of the satellite with respect to the source of light. One of the main aims of this process was checking if the translation it performs internally is accurate enough so that it could be taken as reliable information.

Satellite was tested in different conditions in which the source of the light was moving, while some sensors were blocked. The measurements taken from the GUI were compared with the different attitudes that were commanded to the spacecraft and they coincided in all cases.

Since the response of the system to the different test cases was accurate enough, the information required to perform the PID controller action will be the right one, therefore the retrieved commands of the self-controlling function will be the proper ones.

3.2 Characterization of the ADCS actuator

The actuator of the system is composed by the wheel that allows the rotation of the satellite plus the engines that drives it. As explained before, the wheel is commanded by Pulse Width Modulation(PWM) i.e, a modulation technique, used to encode a message into a pulsing signal. Its main use is to allow the control of the power supplied to electrical devices, especially to inertial loads such as motors.

The average value of voltage (and current) provided is controlled by turning the switch on and off at a fast rate. The total power supplied is proportional to the time that the switch is turned on

The main advantage of PWM is that it is a digital technique easy to implement while the power loss in the switching devices is very low. When a switch is off there is practically no current, and when it is on and power is being transferred to the load, there is almost no voltage drop across the switch. Power loss, being the product of voltage and current, is thus in both cases close to zero.

EyasSat receives a command in terms of revolutions per second, coming either from the user or the self-controlling functions, translates it into PWM following a linear relationship defined as:

$$PWM = a + g * RPS \tag{3.1}$$

This equation can be fully defined by the user by modifying the three controlling parameters, the intercept, the slope and the revolutions. However, there are some limitations in the commanded RPS that can be given to the reaction wheel. The commanded RPS cannot exceed 50 revolutions per second as input, since the system does not admit a greater value for safety reasons. Besides, the reaction wheel should be commanded always with a value of RPS that could provide at least a PWM value of 11.1, which actually it used to be around 20 RPS, in order to move. This limitation arises from the frictional torque that should be overpassed and it will be explained later on.

Whenever the commanded RPS lay within the operational interval the wheel will move, however, the problem arises when the commanded RPS that goes into the formula results in some output RPS that are different from those expected by the user. Some empirical measurements were taken using a tachometer, in order to discretize the system and the response of the PWM. Those results can be found below. Besides, the manufacturing company was asked about the most accurate solution for the modulation they implemented for the PWM system. They offered a discretization of the parameters that makes the resulting RPS follows pretty well the expected results. They affirmed the proper configuration was establishing the intercept to 24 and the slope to 0.4 however that configuration strongly depends on the friction characteristics of each individual wheel. Different configurations were used to obtain different results in order to compare their effectiveness. Actually the comparison shown below is defined for setting the intercept to 0 and the slope to 1, varying then the desired revolutions. In Figure 3.1 there is performed a comparison between the expected RPS commanded to the satellite, the RPS that EyasSat outputs, and also the readings that were made manually using tachometer.

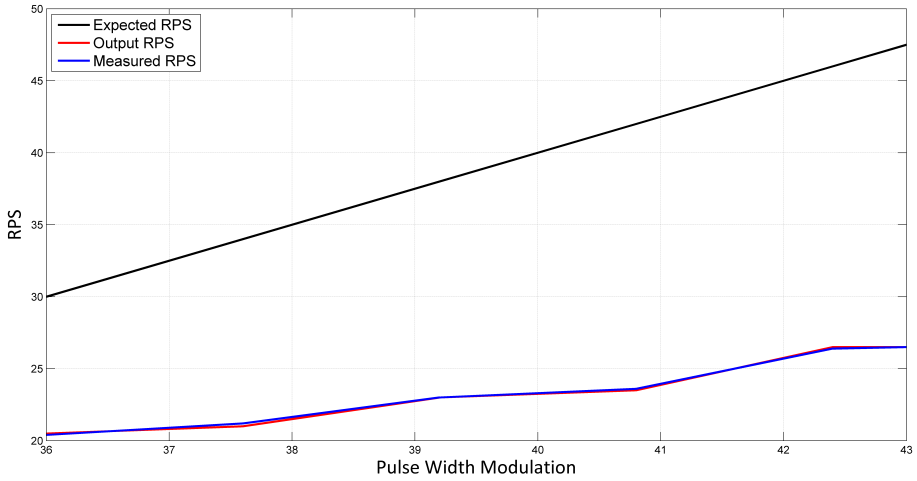


Figure 3.1: Comparison between theoretical and measured RPS

Clearly, the readings of the EyasSat were pretty accurate since the are almost equal as the measurements performed, however, it is clearly appreciable how the controller of the RPS and PWM is not quite well designed.

The difference between the expected RPS and the real ones is too significant, trying to understand it, some different measurements were taken just modifying the parameters a and g, which is the purpose of the Figure 3.2. In Figure 3.2 the reader can find a comparison between the *ideal* configuration provided by the manufacturer company and a random configuration that was implemented.

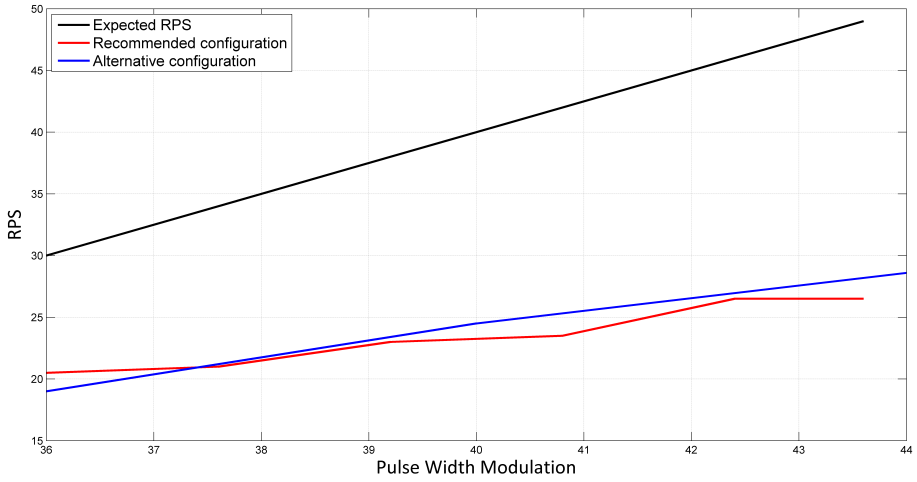


Figure 3.2: Comparison between different configurations of the control

Again as before, the expected result for the RPS is far away from what the user could expect, however it is much more significant how close are both configurations, considering one of them should be optimized and the other not. Therefore, the problem is not the configuration of the controlling parameters, whose output is pretty similar for different cases, whether the gap that does not let the configuration reach the desired behavior.

The difference between the expected curve and the real ones, is pretty similar in both cases. This gap between the curves is due to the rotational friction that appears in the junction of the wheel. Besides, the slope of the real graph is less than the expected one. This performance can be explained since the total torque can be computed as follows:

$$\tau_{net} = I\alpha - \tau_f = I\,\frac{d\omega}{dt} - \tau_f \text{ since the angular acceleration is defined as: } \alpha = \frac{d\omega}{dt} \quad (3.2)$$

However, taking a deeper look into the graphs can be extracted the actual behavior of the reaction wheel is characterized for having a threshold corresponding to the minimum RPS that should be commanded. Hence, the real relationship between the frictional torque and the velocity is $\tau_f = C_o + C_1\omega$. Some experiments were taken using the tachometer for defining the minimum value of PWM that starts the motion of the reaction wheel that has being commented before.

Following the theoretical approach, the greater the angular velocity, the greater will be the frictional torque that the wheel should overcome. Since PWM is used to maintain the revolutions as constant as possible, the gap will increase as it does the ω_{wheel} as seen in the previous graphs. The only way to overpass the gap would be modifying the parameters a and g in such a way the real behavior emulates the expected, however there are some physical limitations of the reaction wheel that make it not able of following that trend.

3.3 Satellite tensor of inertia

One of the most critical parameters that is involved in the self-controlling functions is the inertia tensor of the different components of the spacecraft. Because the wheel control, and hence satellite control, are mainly defined by its inertial properties. However since the EyasSat will be hang out from a cable while the standalone operations, the only term that really maters is the vertical component of its tensor as it can only rotates in one direction. This term comes from the definition of angular momentum where one can compute its value through the torque, by applying:

$$\frac{d\bar{H}}{dt} = \bar{\tau} \quad (3.3)$$

Where the angular momentum can be split as the addition of both contributions the one coming from the wheel plus the one coming from the spacecraft i.e,

$$\bar{H} = \bar{H}_{sat} + \bar{H}_{wheel}. \quad (3.4)$$

Since each contribution can be expressed as $\bar{H} = \bar{\bar{I}}\bar{\omega}$, keeping in mind the angular velocity is a vector, and assuming the torque will remain constant either zero i.e, derivative will be null in any case, the final equation will be:

$$\bar{\bar{I}}(\frac{d\bar{\omega}}{dt})|_{sc} + \bar{\bar{I}}_r[\frac{d(\bar{\omega} + \bar{\omega_r})}{dt}] = 0. \quad (3.5)$$

Where, for simplicity, the properties related to the spacecraft has no sub index whereas the ones related to the reaction wheel has an "r". It is worth to mention that the angular momentum of the spacecraft is only affected by its velocity; however in the later terms related to the wheel it is appreciable how the angular momentum of the wheel is affected by both velocities.

Since the term called ω_r is the relative velocity of the wheel with respect to the satellite, then the total angular velocity would be $\omega + \omega_r$. Equating both terms it is obtained that:

$$(I + I_r)\frac{d\omega}{dt} = -I_r\frac{d\omega_r}{dt} \tag{3.6}$$

This is the equation that is used by the self-controlling equations that are developed in Chapter 5 to adjust the velocity that should be commanded to the satellite. Clearly, the final effect in the satellite will be greatly affected by the ratio $\frac{\bar{\bar{I}}_r}{\bar{\bar{I}}+\bar{\bar{I}}_r}$.

The next step is computing all the contributions that are added by the different parts that compose the final spacecraft. All the components are treated like flat prisms, where every surface is continuous, just to simplify the computation. There are two kind of shapes that should be considered, they are the boards and the sensor. All the boards have the same shape and can be treated like solid cuboid whose inertia tensor is defined to be:

$$\begin{bmatrix} \frac{1}{12}m(b^2+c^2) & 0 & 0 \\ 0 & \frac{1}{12}m(a^2+c^2) & 0 \\ 0 & 0 & \frac{1}{12}(a^2+b^2) \end{bmatrix} [kg/m^2] \tag{3.7}$$

Where the different coordinates a, b and c are the width the depth and the height respectively.

On the other hand the sensor and the wheel, which have the same dimensions, can be treated as solid cylinders, whose inertia tensor is computed like:

$$\begin{bmatrix} \frac{1}{12}m(3r^2+h^2) & 0 & 0 \\ 0 & \frac{1}{12}m(3r^2+h^2) & 0 \\ 0 & 0 & \frac{1}{2}mr^2 \end{bmatrix} [kg/m^2] \tag{3.8}$$

After weighting each component, using this approach allows the user to have an accurate measure about the inertial ratio just using simple formulas. All the tensors are reduced just to the diagonal since all the inertia products are null due to the symmetry planes that are considered in each figure. The whole tensor has been computed in order to have fully knowledge about the system although only matters the component along the vertical direction.

The different dimensions of the composing parts will be presented below so that the reader could check the computation of the inertial properties.

- ADCS, COMM & DH boards:

  These three boards have the same dimensions since they should be located one over each other, being assembled as a unique block. They are all shaped as a square, whose sides measure 11.5 cm. The greatest height of the block is 2.3 cm, since this is the height of the central bus. However for the ADCS board, the greatest

height is 6 cm since the wheel is attached to it.

Regarding the weight of each board, using a scale the following numbers were obtained: 270.1 grams for the ADCS considering also the wheel, 45.36 grams for the wheel, hence the board only weights 224.74 gr, 67.5 gr for the COMM board and finally 63.9 gr for the DH.

- Power board:

  The power board is located at the bottom of the assembly and therefore the dimensions should be greater since it should withstand the weight of the system. It is again a square, however its sides measures 19 cm each. Concerning the height now it is 3.7 cm again due to the height of the central bus.

  Since all the buses are always greater in height than the rest of the board, the height that has been taken into account when computing the inertial properties is averaged.

  The power board's weight is much greater since the batteries of the system are located in that part. The scale tells it weights 841 g.

- External structure:

  The casing structure is the combination of three vertical plates plus the top one that carries the sun sensor. The sun sensor is a cylinder whose radius is 4 cm and its height is 2.1 cm. Its weight is 102.3 g. When the sun sensor was unmounted for measuring its properties, the top panel was measured too. That panel has also a squared shape and its sides measures 18.5 cm. The difference found between the top and the bottom panel is due to the thickness of the structure that is exactly 0.5 cm. This is actually the thickness of this top panel. Finally its weight was characterized to be 189 g.

  The other panels that form the structure are identical. Their height is 20.5 cm and 18.5 cm of width. However, as it has been explained before, there are some subsystem installed on them, therefore the thickness of these panels is not 0.5 cm as it could be expected whereas its thickness was taken to be around 1.5 cm for the inertial properties. There is one external panel that does not have any subsystem installed, hence its thickness is already considered to be 0.5 cm.

Once all the components are characterized, the user can know where is the center of gravity of each of them. Since they have being characterized as simple prisms with well-known symmetry planes, the C.O.G computation is pretty easy. It will be useful later on since the user will need to know where is the C.O.G of the whole system. It will be computed using:

$$R = \frac{1}{M} \sum_{i=1}^{n} m_i r_i. \tag{3.9}$$

Where the"R" means the distance until the total C.O.G from a predefined point of reference, the "M" represents the total mass of the system and the lower letters "m" and "r" are the mass and the distance from the reference point, respectively, of each of the C.O.G (i) that are considered in the whole body.

After that, the only task remaining is the addition of all the contributions. It can be done easily just by applying Steiner's theorem. Steiner's theorem says that the

inertia tensor of a body can be translated to other point different than its C.O.G just by adding to it the product between its mass and its distance squared. Expressed in a vectorial form, taking the C.O.G as the base of the vectorial space:

$$\bar{\bar{I}}_p = \bar{\bar{I}}_G + m\bar{d} \otimes \bar{d}. \tag{3.10}$$

Where the letter "P" indicates the point in which the tensor will be known, "G" represents the C.O.G and $\bar{d}$ is the vector that joints both points in the vectorial space.

Using Steiner's theorem allows the user to characterized some parts that have different shaped components like the top panel of the satellite. Here the sensor is joined to the prism. Following this approach, the whole body is finally simplified to 9 components of properties totally known.

Finally, to compute the total inertia tensor of the satellite, the user only has to translate the properties of these 9 components from its COG until the total C.O.G of the complete system through the application of the Steiner's theorem again.

The final tensor computed once all the contributions have been considered is :

$$\begin{bmatrix} 0.0100 & 0 & 0 \\ 0 & 0.0143 & 0 \\ 0 & 0 & 0.0168 \end{bmatrix} [kg/m^2] \tag{3.11}$$

Hence, the vertical component of the satellite is $I_z = 0.0168\, kg/m^2$, whereas the z-component of the inertia tensor of the wheel is $I_{z,wheel} = 9.3621 * 10^{-5}\, kg/m^2$.

Reader should notice that $\frac{I_{z,sat}}{I_{z,wheel}}$ is approximately 200 which implies that the wheel should rotate much faster in order to move properly the satellite.

Chapter 4

Emulating the system through Matlab simulators

4.1 Adjustable PID law

Since the control performed over the reaction wheel is not too accurate, as seen in Chapter 3, a theoretical analysis about using a "Proportional Integral Derivative" controller has been applied to the system. This controller, also known as "PID" controller should be explained: Proportional-Integral-Derivative (PID) control is the most common control algorithm used in industry, its popularity can be attributed partly to their robust performance in a wide range of operating conditions and partly to their functional simplicity, which allows engineers to operate them in a simple, straightforward manner.

As the name suggests, it consists of three basic coefficients; proportional, integral and derivative which are varied to get optimal response. They are called gains and they are represented with a K as can be appreciated in Figure 4.1.

The basic idea behind a PID controller is to read a sensor, then compute the desired actuator output by calculating proportional, integral, and derivative responses and summing those three components to compute the output. A scheme of the process can be seen in the Figure 4.1.

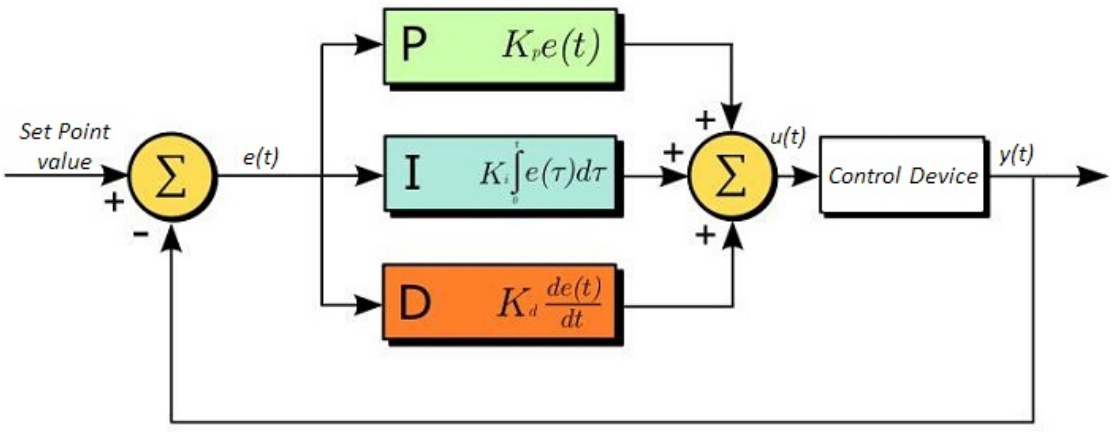


Figure 4.1: Graphical representation of a PID controller in closed-loop configuration

The behavior of the system will differ depending on which gain is used into the system, and so the response:

- Proportional Response: The proportional component depends only on the difference between the set point and the process variable. This difference is referred to as the Error term. The proportional gain (K_p) determines the ratio of output response to the error signal. For instance, if the error term has a magnitude of 10, a proportional gain of 5 would produce a proportional response of 50. In general, increasing the proportional gain will increase the speed of the control system response. However, if the proportional gain is too large, the process variable will begin to oscillate. If K_p is increased further, the oscillations will become larger and the system will become unstable and may even oscillate out of control.

- Integral Response: The integral component sums the error term over time. The result is that even a small error term will cause the integral component to increase slowly. The integral response will continually increase over time unless the error is zero, so the effect is to drive the Steady-State error to zero. Steady-State error is the final difference between the process variable and set point. However, since the integral term responds to accumulated errors from the past, it can cause the present value to overshoot the setpoint value.

- Derivative Response: The derivative action predicts system behavior and thus improves settling time and stability of the system. That component causes the output to decrease if the process variable is increasing rapidly. The derivative response is proportional to the rate of change of the process variable. Increasing the derivative time (Td) parameter will cause the control system to react more strongly to changes in the error term and will increase the speed of the overall control system response. Most practical control systems use very small derivative time (Td), because the Derivative Response is highly sensitive to noise in the process variable signal. If the sensor feedback signal is noisy or if the control loop rate is too slow, the derivative response can make the control system unstable.

Then control law will be then:

$$u(t) = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{d}{dt} e(t) \tag{4.1}$$

Following these approach, a physical representation of the system can be done, just to understand how the approach must be developed. For any physical system in which an engine-driven rotational device is involved, the control variable will be the Torque generated by its engine. However, the selected setpoint, at least for that case, will be the rotational position at which it is set at any instant of time. Knowing this characteristics, the physical representation of the system is developed an can be seen in Figure 4.2.

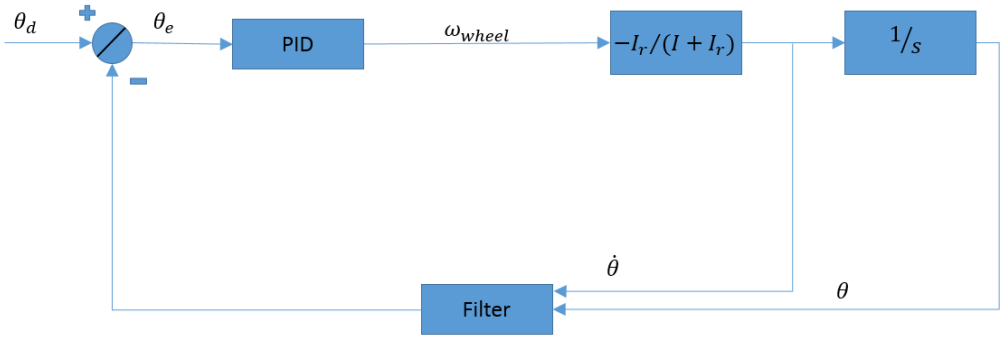


Figure 4.2: Physical scheme of the system in closed loop configuration.[1]

The required angular velocity of the wheel is computed into the PID block. The following step is computing the angular velocity that should be commanded to the satellite in order to reach the desired setpoint. The box containing the term $\frac{1}{s}$ is called "integrator block", it is the equivalent to an integral in Laplace domain. Finally, this information about the rotational position and velocity will define the state of the system that enters into the box called "filter". This box represents the sensors and internal computations that are carried on into the EyasSat, closing the loop.

Knowing how to deal with the physical variables that are related during the whole process, the theoretical development of the physics of the problem will be performed. The starting point is the application of the angular momentum conservation law to the spacecraft during its operation, this leads to:

$$\bar{\bar{I}}(\frac{d\bar{\omega}}{dt})|_{sc} + \bar{\bar{I}}_r[\frac{d(\bar{\omega}+\bar{\omega_r})}{dt}] = 0. \tag{4.2}$$

which being integrated gives:

$$(I + I_r) \times \bar{\omega} + I_r \times \bar{\omega_r} = 0 \tag{4.3}$$

The next step is translating the variables into Laplace domain. Then, the satellite rotation called θ will be obtained as $\theta = \frac{\omega}{s}$, which is the Laplacian operation for the integration. It is worth to mention that the initial conditions of the functions are considered to be zero, for simplicity and this is why they are not appearing in the final result.

The next step would be obtaining the control parameter in terms of the different gains and the error of the system. The usual control parameter in a system like that, would be the torque that is provided to the wheel through the engine installed on it. However, this kind of control is the one used by the algorithm of the original software and cannot be overpassed by users using any kind of software. Therefore the control

[1] θ_d and θ_e are the setpoint and the error of the system respectively.
ω_{wheel} is the angular velocity of the satellite
θ and $\dot{\theta}$ are the state variables of the satellite.

theory analysis is reduced to the parameters that concern the velocity and the orientation since the only controlling actions that can be applied are on the commanded RPS of the reaction wheel. This fact will greatly affect the results of the following analysis.

Due to the fact that the control parameter of the system it is actually the commanded velocity of the reaction wheel, the control law will be written as

$$\bar{\omega_r} = (K_p + sK_d + \frac{K_i}{s})\bar{\theta_\epsilon} \tag{4.4}$$

where the variable called $\bar{\theta_\epsilon}$ is the error function of the system, which is defined as $\bar{\theta_\epsilon} = \bar{\theta_d} - \bar{\theta}$.

Joining the results obtained until now, the following equation is obtained:

$$\bar{\theta} = \frac{\omega}{s} = -\frac{1}{s}\frac{I_r}{I + I_r}(K_p + sK_d + \frac{K_i}{s})(\bar{\theta_d} - \bar{\theta}) \tag{4.5}$$

Then, taking the variable $\bar{\theta}$ as a common factor the resulting equation appears:

$$\bar{\theta}[1 - \frac{I_r}{I + I_r}\frac{1}{s}(K_p + sK_d + \frac{K_i}{s})] = -\frac{\bar{\theta_d} I_r}{s(I + I_r)}(K_p + sK_d + \frac{K_i}{s}) \tag{4.6}$$

Now, the principles of control theory will be used in order to analyze the stability of the system. The transfer function is required for the analysis. Strictly speaking, it is a representation in terms of spatial or temporal frequency, of the relation between the input and output of a linear time-invariant system and actually in the system it can be represented as:

$$G(s) = \frac{\bar{\theta}}{\bar{\theta_d}} = \frac{-I_r}{s(I + I_r)}\frac{(K_p + sK_d + \frac{K_i}{s})}{[1 - \frac{I_r}{I+I_r}\frac{1}{s}(K_p + sK_d + \frac{K_i}{s})]} \tag{4.7}$$

which being developed will lead to

$$G(s) = \frac{-I_r(s^2 K_d + sK_p + K_i)}{s^2[I + I_r(1 - K_d)] - I_r s K_p - I_r K_i} \tag{4.8}$$

The analysis of the stability of the system is based on the roots of the denominator of the transfer function. For that, the following lineal equation of second order should be analyzed:

$$DG(s) = s^2[I + I_r(1 - K_d)] - I_r K_p s - I_r K_i \tag{4.9}$$

Before developing the whole analysis of the system, some assumptions should be taken and they should be justified. As it has been commented above, the control theory that is applied to the system should be characterized since no one can overpass the original PID that the company embedded on the spacecraft.

Strictly speaking, the control theory says that the function that should be used for that problem is

$$\tau(t) = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{d}{dt} e(t) \tag{4.10}$$

However, since the state variable that can be controlled by the user is the rotational velocity instead of the torque the equation should be modified knowing that the torque can be expressed as: $\tau_{net} = I\frac{d\omega}{dt}$, which being integrated yields to:

$$\int_t^{t+dt} \tau_{net} = I[\omega(0) + \omega(t)] \tag{4.11}$$

Finally, asumming zero initial conditions for simplycity and knowing the control parameter will be the rotational velocity, the final equation of the analysis will be:

$$\omega_r = K_p \int_0^t e(t)dt + K_i \iint_0^t e(t)dt + K_d \int_0^t \frac{d}{dt} e(t)dt \tag{4.12}$$

Where the inertial term has been included into the constants in order to simplify the final equation. Seeing that equation, the user can deduce that make no sense including the integral gain in the analysis since there is no a physical meaning for that double integration. Actually, the proportional gain becomes an integrated gain and the derivative term becomes a proportional one.

Due to this fact, the integral gain is directly set as zero in the self-controlling functions. The definition of the two other gains will be described in the following section.

4.2 Adjustment of PD gains

The adjustment of the proportional and the derivative gains is basically performed by tuning the system by hand. In order to asses how the system will response when some controlling actions are implemented into it, some simulators of the system were created. That simulator creates some initial conditions concerning the state variables of the system, i.e, its orientation, its velocity, its set point, etc. and it computes how the satellite will behave until reaching the specified set point. Following that approach, the user can have some information about the real behavior of the system prior to modify any physical characteristic of the real model. To perform that simulation, different models were created.

4.2.1 Matlab's PD controlled simulator

The simulator developed with Matlab will be the tool used to predict the response of the system and tune the gains. For the tuning of the system, the user has available an .m file that can run, in which an inner function sets different initial conditions and different setpoints to the satellite. These parameters, once known, are sent to other functions that simulates the behavior of the satellite, create an input to the sensors, translate them into a real angle, compute the required velocity and solve the problem so that the satellite could reach the final setpoint after some iterations. This simulator was created early, so that the working principles of the PID controller could be known properly before starting any physical modification.

After some trial using the simulator, the tuning of the PID was done. There are two study cases in which the system had to be tuned. The easy and first one, is that the angular momentum that is exerted over the satellite due to the environment and fixing system is zero. The other one is that it is constant but different from zero.

For performing the tuning different gains were tested and the response of the system was computed in each case in order to compare them. Besides, the velocity commanded to the wheel was also computed, just to check if the torque required for each gain could be exerted by the driven engine.

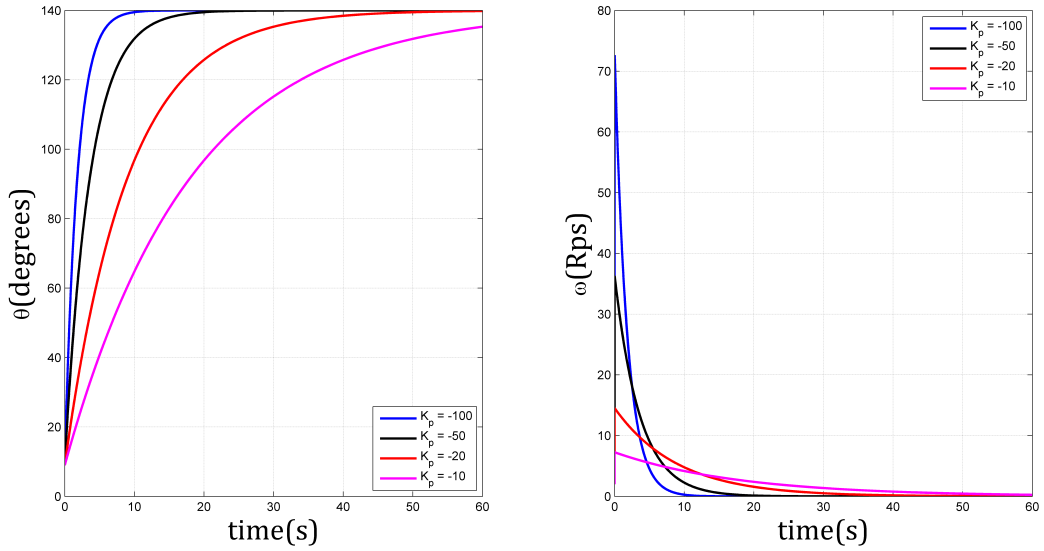


Figure 4.3: Tuning of the system by simulating EyasSat's behavior, from $\theta = 9^{o}$ up to $\theta = 140^{o}$

Seeing the different behaviors of the system and how much torque, and therefore effort, each configuration would cost, it can be said that any desired position can be reached easily for every gain. Although setting it to $K_p = -10$ would lead to a slow response, the other three configurations are quick enough. However, checking the angular rotation of the satellite that is required to reach that configuration, it can be easily seen how the two most aggressive configurations, i.e, $K_p = -100$ and $K_p = -50$, require a huge amount of angular rotation of the wheel and perhaps it could overstep the maximum velocity of the reaction wheel. Hence, that configurations would required a huge Torque to be exerted by the engine. Since the maximum torque of the wheel is not known, although that information was requested to the manufacturing company, the appropriate gain seems to be the value of $K_p = -20$. The reason for that is because the satellite can reach its desired position, without risking the wheel by requiring too much angular velocity, and therefore to much torque, compared to the one that should be required for greater gains.

Furthermore, there could be other cases in which the satellite cannot response as expected, and therefore they should be checked. The concerned case refers to the saturated conditions that are encountered in real spacecrafts. This condition can be critical in a satellite and it is worth to be checked. Whenever the wheel starts to accumulate angular momentum and cannot be released, finally they saturate as they are still operating until they finally lose satellite's control. For checking that situation,

the simulator has a velocity limiter implemented on it, so that it saturates the wheel whenever it exceeds the limit.

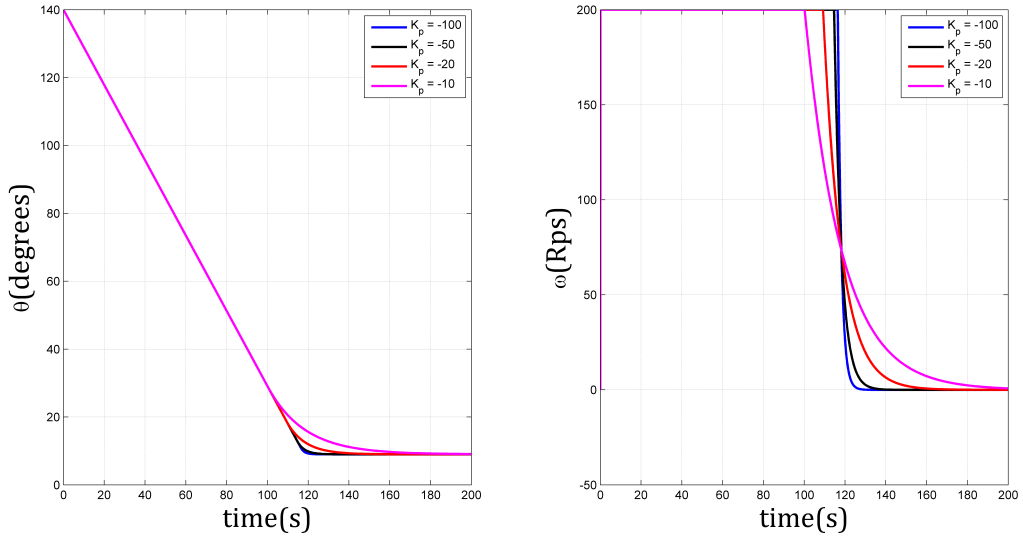


Figure 4.4: Simulation of EyasSat's behavior when saturated

Again, the resulting behaviors show the configuration of $K_p = -20$ can produce the required response without requiring aggressive modifications in the velocity, i.e, without requiring excessive values of the wheel's velocity. It is worth to mention how the saturation of the system modifies the role of the proportional gain. Whereas in current conditions the proportional gain is the key parameter, in saturated conditions it is only relevant in the neighborhood of the commanded angle.

Since there is no a great advance in increasing the proportional gain, although there is a demand for the torque the selected configuration is using $K_p = -20$. Besides, since any setpoint can be reached just by using the proportional gain, the derivative gain is selected to be zero, in order to simplify the controller.

4.2.2 Bang-Bang simulation

There were other simulations created apart from the one explained above. In the first case, the satellite has not any register about its previous position or the way it has behaved before, as it happens in a pure proportional controller. It means the satellite does not carry information about previous states of the variables neither takes into account the future behaviour. In this approach the satellite only recognizes where it is and where it should go and instantaneously it commands an adequate velocity to itself. In the next iteration it just checks where it is and gives another instruction just considering the difference between the state variable and the setpoint. This kind of controller is called bang-bang or deadband control, since it cannot never reach the appropiate setpoint. This happens because it only actuates when the system moves out of the deadband, therefore it is kept always oscillating around it. This inability to reach the adequate setpoint is due to the absence of the derivative and the integral gains.

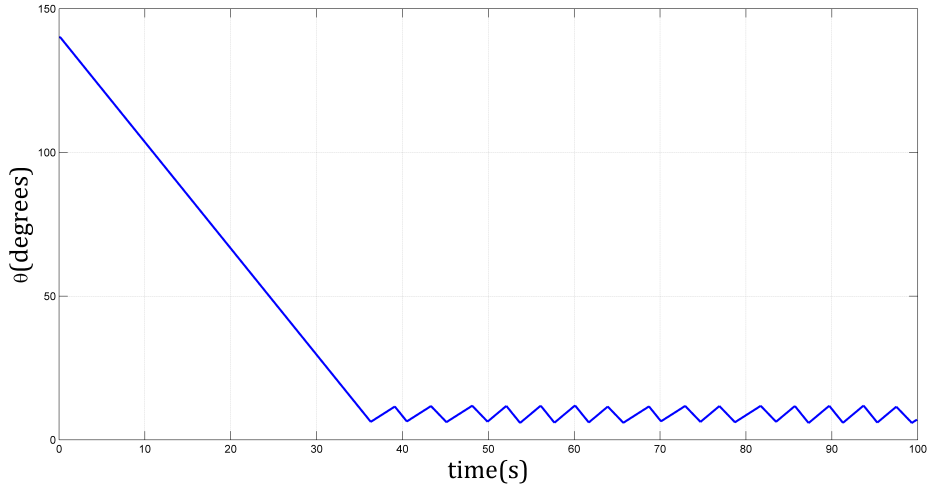


Figure 4.5: Bang-Bang simulation of angular position

Right below, the reader could check the behavior of the satellite and see how the velocity looks like to be saturated as in the previous analysis. There is a zoomed in view in order to ease the sight of the graph.

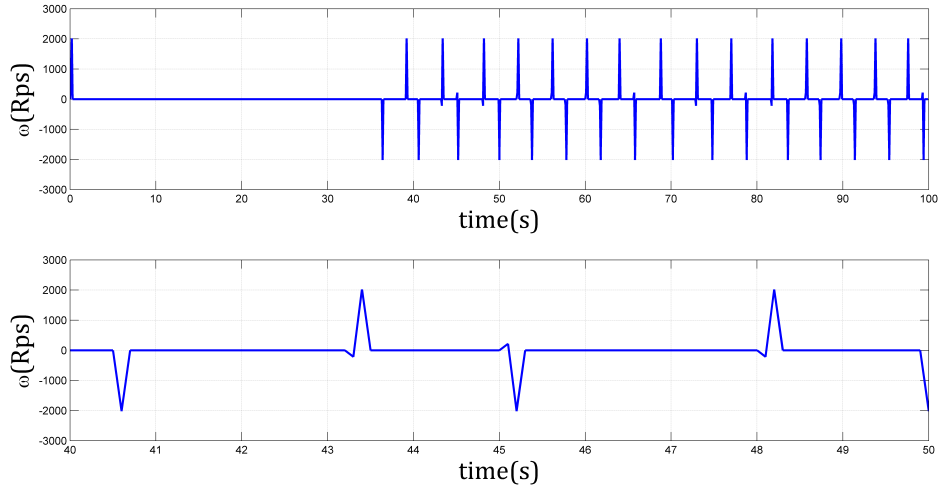
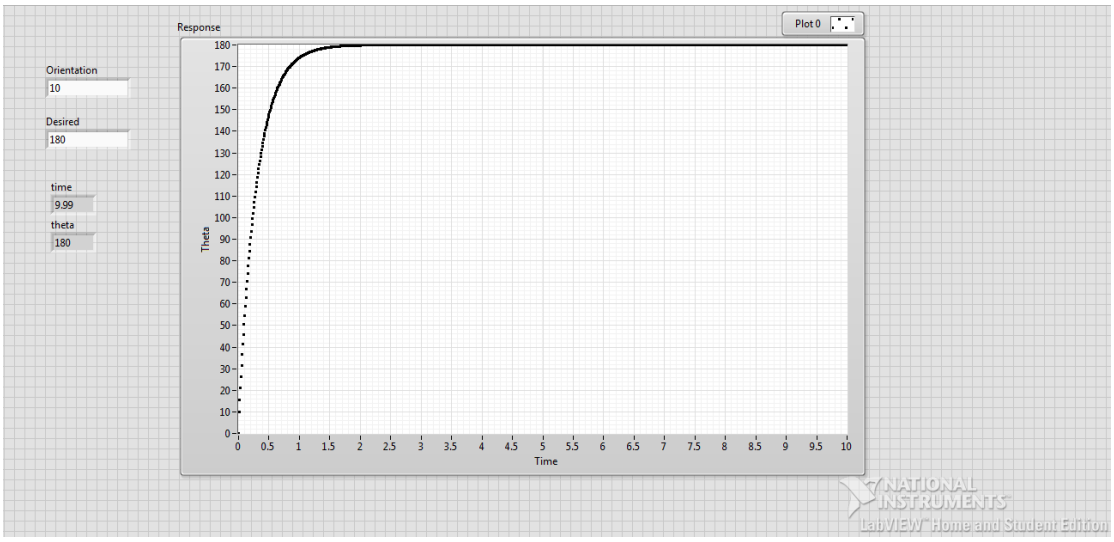


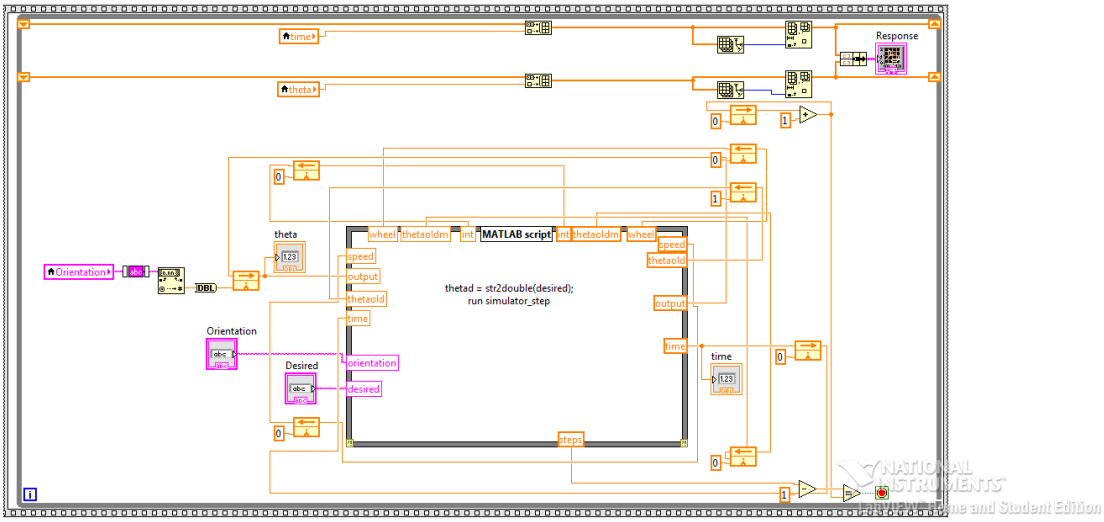
Figure 4.6: Bang-Bang simulation for angular velocity

4.2.3 Other simulators

Some of these models have been developed using Labview instead of Matlab, just for comparing both methods. Below in Figure 4.7 there is a representation of the simulator.



(a) Labview simulator's GUI



(b) Labview simulator's source code

Figure 4.7: Simulator developed in Labview

Chapter 5

EyasSat-Labview-Matlab interface

5.1 Interface capabilities

The GUI allows the user to have, in a quick view, the complete information related to any variable that retrieves the satellite. In the GUI the user will find also the buttons that allow to choose between the different functionality of the code. There are some boxes adjacent to the buttons that should be explained. Two of them are writable cells in which the user can define the desired angle in which the satellite should be, and also the command the satellite should read. There is a third box in which the user can read the communication port that is being used by the ground station. Finally, the last and fourth cell is created just for letting the user check if the satellite is answering in a proper way. It represents the exact velocity that the PID self-controlling function is commanding to the wheel. This information could be read in the variable called *RPS_out*. However, having this extra cell reduces the time needed to check the commanding velocity, since it appears immediately instead of waiting until the next cycle. Besides it permits the user to check if there is any problem in the communication between the GS and the satellite. These graphical tools that are available for the users can be seen in Figure 5.1.

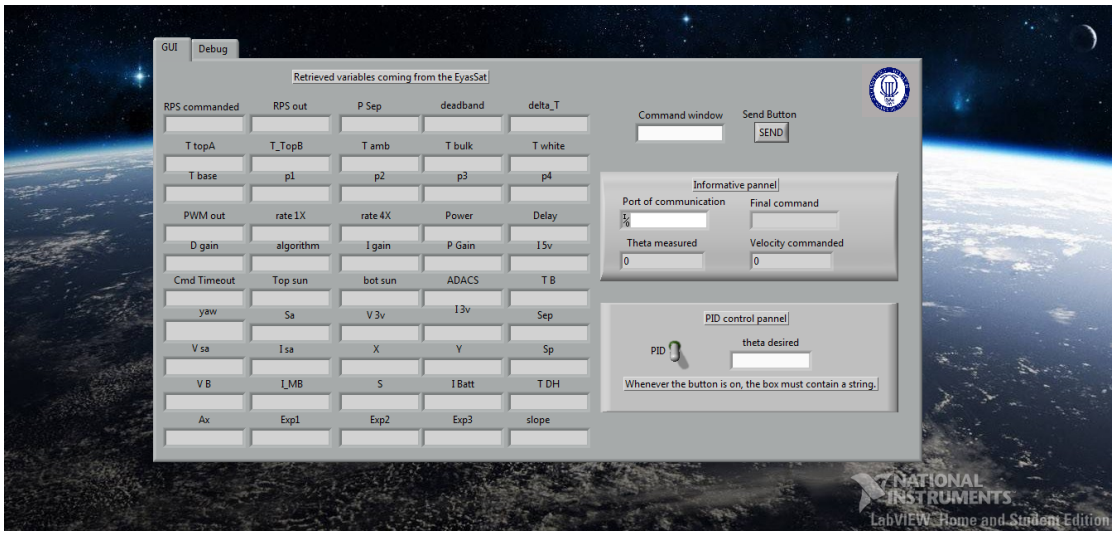


Figure 5.1: GUI developed for begginers

Additionally, it shows in the second tab, which can be seen in Figure 5.2, the complete message that the satellite is emitting. In that way, any advance user can check what happens in case any error arises in the process. The lack of information due to some unconnected boards, non-varying properties of the telemetry or even discordance between the data that should be read and the one shown are the main problems that raised during the work and it is worth to be advised. The code is already prepared to face them but they could appear in case any beginner user modifies some properties.

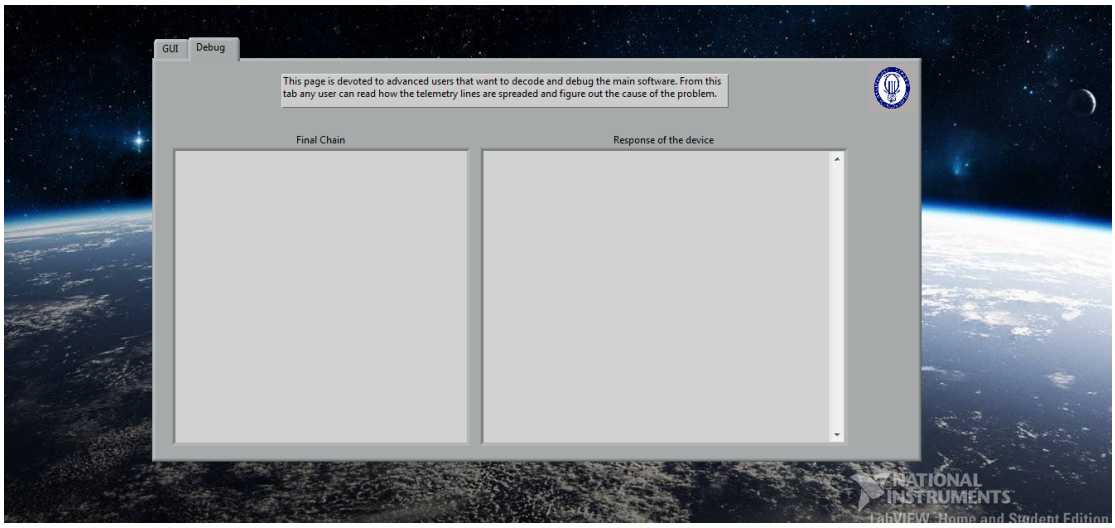


Figure 5.2: GUI developed for advanced users

5.2 General description of the process

The development of the internal process of the final GUI is a key process for the work. These are the different steps that characterized the internal operations:

- First of all, the communication between the "ground station" and the satellite should be established in the proper way. For that purpose, the software needs to choose the port in which the antenna is installed, cleans the buffer in case any undesired information were stored previously and initializes the device. Once that communication is established, satellite can be read so that all the information it retrieves can be obtained, and also it can be commanded, if desired. However, since the communication channel is half-duplex as explained before, there is some limitation about the flow of information and it could imply some delay in the process.
- The following step that is performed internally is choosing the mode in which the user wants to work. Depending on the purposes of the user it can choose between just checking the EyasSat's readings of its state variables, commanding it directly through its language or activating the "autopilot" mode, i.e, the self-controlling functions will handle the information and will command the satellite independently.
- Depending on the user selection, the process will follow one path or the other, there are three options:

    1. In case the user chooses just to command the satellite by itself, it will take the instruction given by the user, it will interpret it and later on, it will be

applied to the satellite in case it matches its own language; after that, the process will jump until the final step and finishes.

2. If the user chooses to read the satellite, the process will recognizes all the information retrieved by the different sensors of the satellite and it will be shown through the GUI.
3. Finally, if the user chooses the self-controlling functions, the satellite will take the information retrieved by the photosensors in order to know its attitude. Following, it applies the P Closed-Loop control to compute the required velocity to achieve the final position and modifies the velocity of its reaction wheel until the satellite is properly oriented. Once the final position is achieved EyasSat is commanded to stay there, until the desired angular position is modified.

- Finally, when the satellite has performed the expected actions, the internal process will reach the final step in which the buffer is cleaned up in order to facilitate later operations. Once this operation is accomplished, the port will be closed, to avoid any possible conflict in case some user tries to access the satellite from a different way while it has its ports opened.

All these steps are divided in different frames of a "flat structure", this is how LabView calls it, that is characterized by setting an inner time line in the process. Every operation that is inside the frame should be accomplished before the code moves onto the next frame. Although this kind of structures can make the process slower and excessively simple if some operations should be performed in parallel, they are a good choice for the purpose of this code in which the only requirement is the clarification of the process and the adequate timing of the steps.

During the creation of the self-controlling functions, the main problem was the integration of the functions within the inner software of the satellite. Since the functionality offered by both software, Matlab and Labview, are quite essential for the development of the project, integration was one of the main challenges as one of the main objectives.

Matlab is a powerful software that allows the user to deal with a big amount of data and processing operations. It offers great advantages when applying the control algorithm, but also when dealing with the data retrieved by the satellite.

Labview is a software developed for helping engineers to design, create, code and produce some graphical interfaces that are able to interact with, collect data from and command different mechanisms or systems. It is worth to mention that Labview is the main channel between the user and the system, making possible the communications and also the commanding of the satellite. Actually, it offers some functions that allow the user to write some Matlab code within a script or running some predefined ".m" files that should be located in the appropriate folder(by default, the main folder that Matlab creates during its installation).

Once the process is understood and collaboration has been established between the different software, the implementation of the process will be explained in the following section.

5.3 Implementation process

Since there are different approaches in which the EyasSat C&CC could be developed, different versions of the final code were created, in order to assess which of them is the most appropriate for the required task. The final version of the code will be shown and explained below. The code, first, let the user choose between writing a command to the wheel or not. Depending on that decision, the code will choose a path to follow that determines the time and actions it should carry on. In case the user decides not to write, it can choose between just reading the sensors to get information, or commanding the system by using the self-controlling P function that is embedded on it. In case it wants to write, the user only has to write the appropiate command and nothing more.

This scheme was finally applied since it was conceived to be the most accurate in order to reduce the processing times. This decision will be easily understood once the whole code is spread out:

The first step is devoted just to initialize the device, the port in which it is connected should be selected. For the radio communication the port is always established as *COM4* whereas for the usual communication it is selected as *COM3*. Then the buffer should be cleaned, to ensure that the reads are in real time and accurate enough. Finally the information related to the selected port and to any arising error is saved in local variables, in order to travel along the whole code, so that this information is available at any moment. This process can be seen below in Figure 5.3.

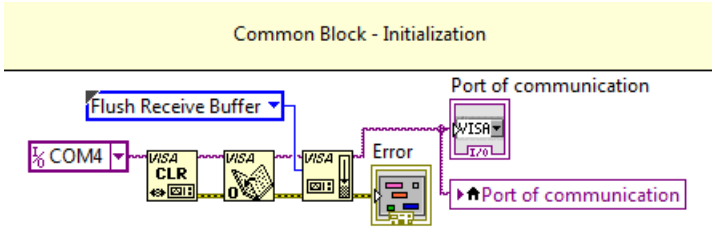


Figure 5.3: Initialization step

In case the user wants to command the satellite in a specific way, it should click a "*SEND*" button. Whenever the code notices this button is pushed, it follows the path seen in Figure 5.4

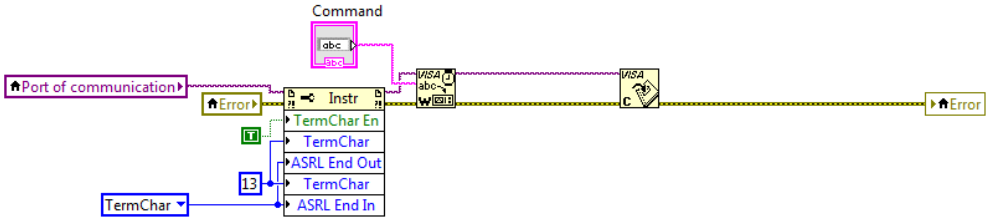


Figure 5.4: Commanding the satellite

The code imports the information concerning the port and the errors, adjust the serial properties that it will use in the writing process and send any array that is written in the *"Comanding"* cell that is in the GUI. After that, it closes the port; this is

done along the whole process to ensure whenever interaction with the satellite will not interfere the following ones. The option that is set in the serial process is the enabling of the termination character. In order to recognize it, Labview should know it could encounter one, and in that case, which one. However, as it has been commented above, there is not a fixed termination character, therefore this step is created to facilitate the post-processing in case it uses the appropriate character. The number *13* defines the character selected is the carriage return. The other chance is using the number *10* which implies a line feed.

Once the message has been delivered and process it by the satellite the code ends and it goes back to the first step.

In case the user does not want to write into the device, whether it wants to use the PID or just reading the sensors, the code will follow another path. The path leads to a common block of functions that will perform its process just once per cycle, reducing in that way the processing time. Below, in Figure 5.5 can be seen where the code starts the device, sets the serial properties of the device and reads from it. Once the information is read, the code saves it in the local variable called *Response of the device*, that will be treated later.

The "SERIAL box" has some properties established that should be explained. The number set to 100000 is the time that it should wait until some information arrives the port. If that time is exceeded and the port remains empty, the code will fail, this is why the number is too huge. The property that is established to "3" sets the termination character. Since the function used here is different than the one used before the character is different but it still refers to the carriage return. Finally there is a 1115 that is linked to the reading function. It defines the number of bits that should be read from the buffer of the device. One telemetry message has exactly 558 bits, therefore the code should exactly take the space that will be occupied by two telemetry messages except for one bit, ensuring in that way that a complete message will be considered although the termination character defined is not the expected one.

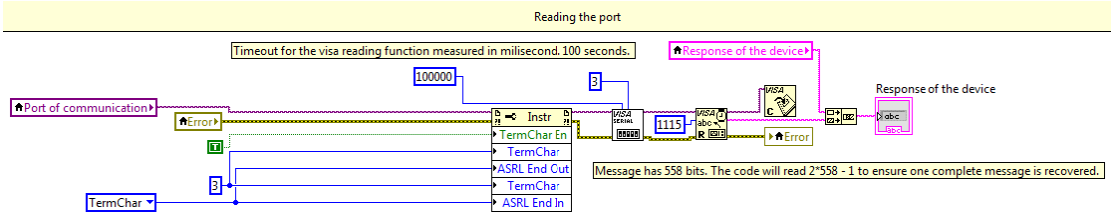


Figure 5.5: Telemetry reading

Whenever the complete message has been retrieved by the satellite the code should process it. Three operations are applied to the message for that purpose. First the code suppress all the white spaces that the message could contain, including line feeds or carriage returns for easy post-processing. Furthermore, it should translate two variables that will bring some conflict due to the similarity in their names. Once all the variables have unique names the message is reduced to the plain text that is included between the terms *Telemetry* and *offset* that are the first and last variables included in one complete telemetry chain. Therefore the code takes just the desired message. This

process can be seen below in Figure 5.6

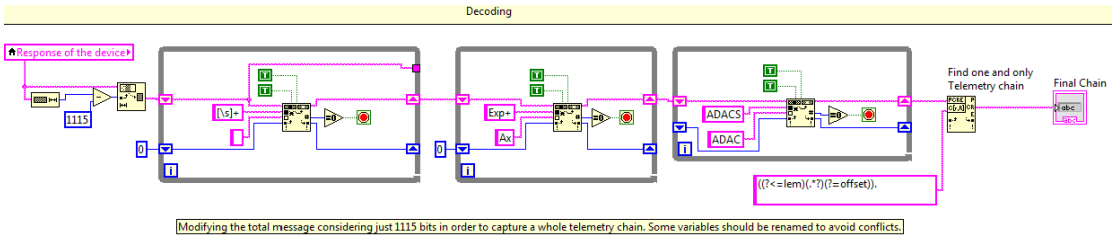


Figure 5.6: Decoding the message

Once the message is simplified, decoded and processed, the code should recognize the information that accompanies any variable, collecting it and presenting it to the user through the GUI. This process is simple but tedious since there are approximately 50 different variables to read, increasing a little bit the characteristic time of the code. Below in Figure 5.7 the reader will find an example of how is the reading of one variable. Whenever a variable cannot be read, because of the omission of that information in the telemetry, the code will notice that and will present again the preceding value, so that the user could have an approximate idea about what is happening to the satellite all the time.

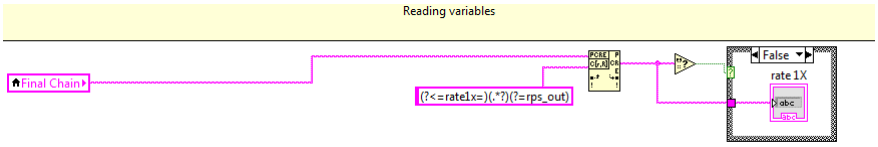


Figure 5.7: Variables obtention

At the moment the common block of processing functions is over, the code goes on, leading the user to another choice. Since the code should know whenever the user wants to read or not, there is another button in which it defines what it wants. If the button is clicked and set to *TRUE*, the PID system is activated. If the button is not modified, the code set it initially to *FALSE* and the code just leaves the frame and starts again.

When the button is set to *TRUE* the PID takes the control of the satellite and starts to command it. The first action that the self-controlling function does is setting the specific parameters that allows it to command the wheel. As it has been explained before, it uses the algorithm "0" that modifies the Pulse Width Modulation value by setting the intercept and the slope of the graph. As can be seen in the Figure 5.8 the intercept is set to 12 and the slope is set to 0.6. Those numbers are not set randomly, since the intercept is set following the manufacturer's recommendations and the slope is just a little bit higher than the recommended one, to ensure a quick response of the wheel.

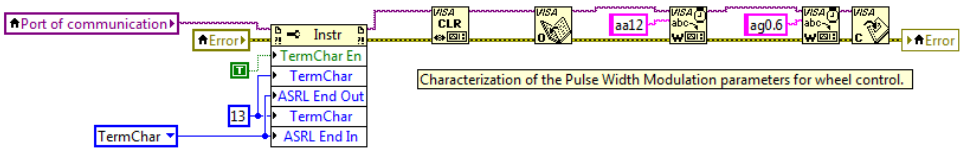


Figure 5.8: Characterization of the wheel

Once the required parameters are correctly established in the satellite's CPU, the self-controlling function takes the reading of the variable called *yaw angle*, computes the distance that should be cover until reaching the desired position, which is input by the user and considering the inertial equilibrium between the system and the wheel computes the required velocity that should be commanded to the wheel. This self-controlling function will be presented together with all the other used functions in annex 2, at the end of this report. The computed velocity is converted to a string and concatenate with the adequate command required, so that the satellite could interpret it correctly. It is important to notice that the satellite will not recognize decimals, however computed velocity is likely to have them, therefore it should be rounded and translated into an integer. The whole process can be seen in Figure 5.9.

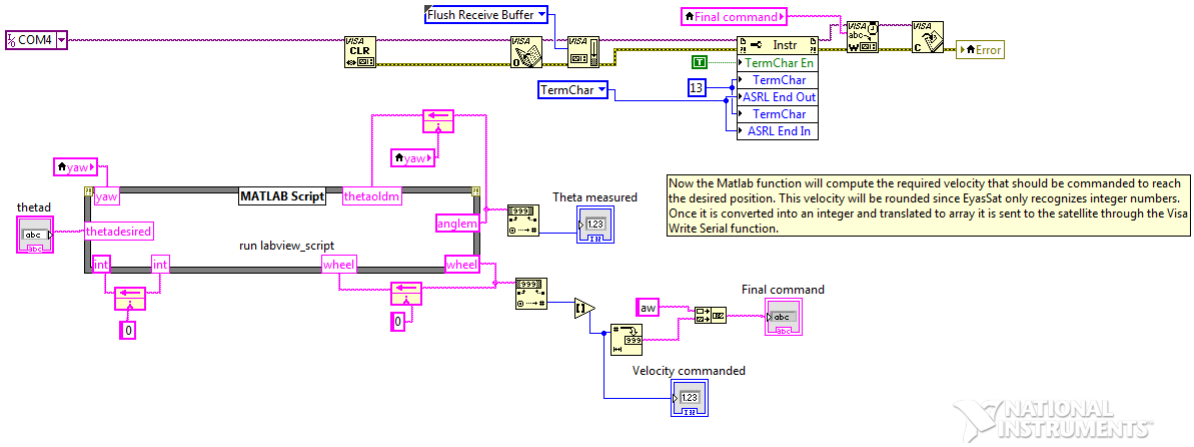


Figure 5.9: Self-controlling function

Finally, when the velocity is commanded to the satellite, the variables concerning the velocity, the time and the position of the satellite are stored for later processing. It will allow some comparisons to determine if the PID system is good enough or if the system can be controlled adequately. The functions that are available in Labview for collecting, storage and saving some variables are not as intuitive as the ones find in Matlab. In Figure 5.10 the reader can see an example.

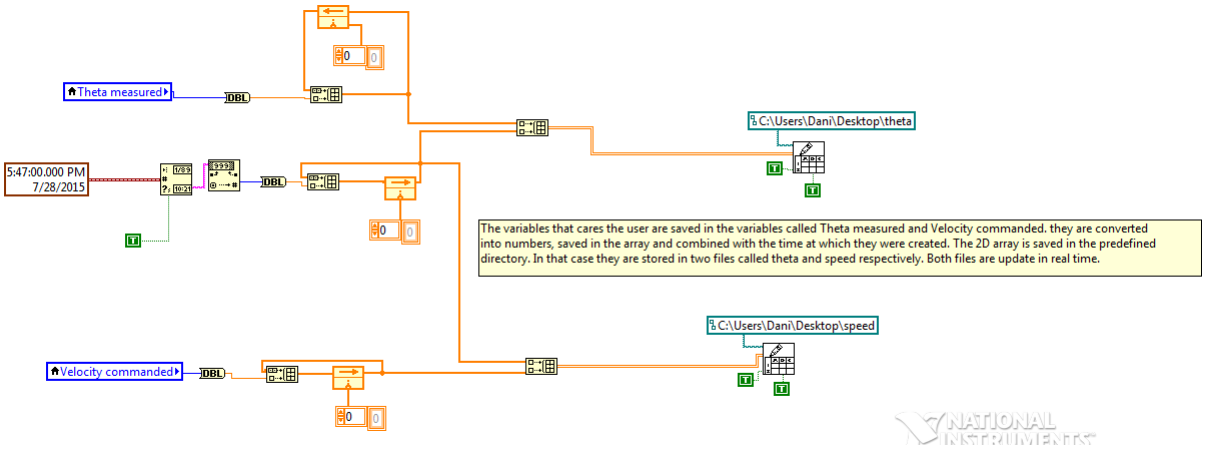


Figure 5.10: Storage of the data

Chapter 6

Control response and performance results

6.1 Response of the system in Open Loop configuration

Knowing the sensors were integrated perfectly with the new software, there should be check if the actuator can be commanded through the GUI and specially if it is commanded in the right way.

The first step of the operation is the experimental set up of the system. For that purpose the satellite was configured as "in-flight operation", hanged out from its support and released as can be seen in Figure 6.1. Communications were established through the radio antenna, to know how the spacecraft is behaving at any time and finally the GSE plug is connected providing the power to the satellite.

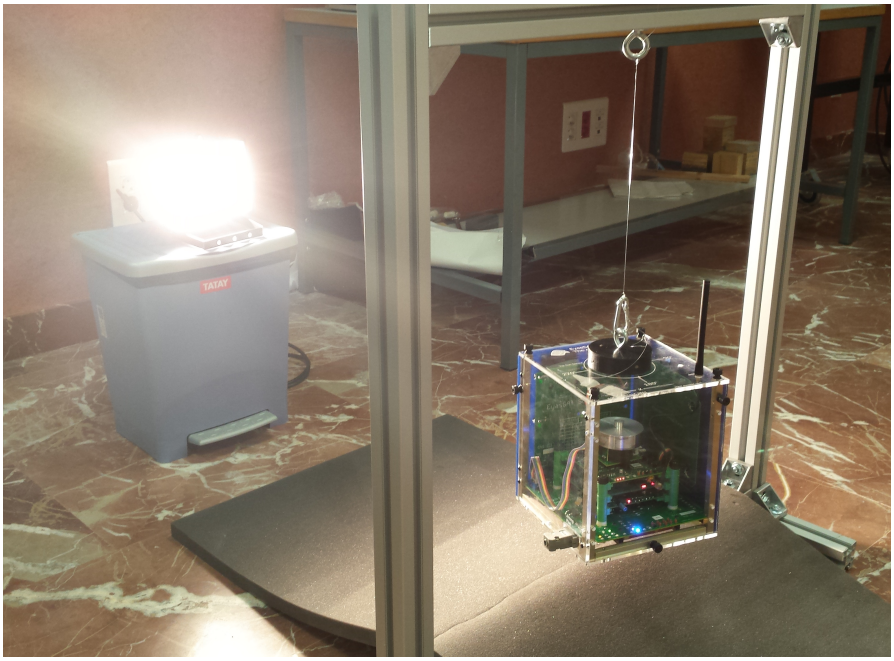


Figure 6.1: Supporting device to emulate "in-flight" conditions.

Once everything is connected, using the C&CC some commands were sent to the satellite. Primary trials of communications were based on some trivial modifications, for instance, the date at which it is established (sending "*ch XX*", "*cm XX*" and "*cs XX*" for the hours, minutes and seconds respectively), the channel used to retrieve information("*is XX*") or the telemetry delay("*id XX*"). Apparently, it response accurate to these modifications. After that, relying the satellite can handle the information and commands properly some actions were sent. The satellite was asked to modify the velocity of its reaction wheel, to stop it and also to modify the rotational direction(all the commanding actions based on modifying the parameters a, g and RPS by sending "*aa XX*", "*ag XX*" and "*aw XX*").

After commanding different actions to the satellite and see how it reacts, it can be said that apparently the satellites reacts properly to the new configuration of the software, since the system could read and interpret the information.

However the characteristic time required for the system to retrieve telemetry, to evaluate the given commands and specially to transmit them to the actuators is excessive. The internal processing of the information to the satellite requires about 5-6 seconds until it process all the data. Besides, the actuator also requires a little bit of time to overcome the frictional torque of the wheel and to reach the commanded velocity.

These results, although they were positive since the satellite could answer to the commands, started to show that maybe the satellite was not appropriate for a PID commanding system, since it requires a complete knowledge of the state variable in real time at any instant of the experiment.

6.2 Response of the system in Closed Loop configuration

In order to see if the excessive characteristic time of the system is also too big for the closed loop operation, another test was performed. For that, the self-controlling function embedded on the PID was activated through the button that is devoted to it in the GUI, and the system started to operate by itself. As expected, the system process properly the environment information and it computes perfectly the yaw angle from which the source of the light was shining, it showed all the properties in real time of the different states of the satellite through the GUI and finally it computed properly the rotational velocity that should be commanded. However, as commented previously the satellite has a real limitation due to the reaction wheel, since the commanded revolutions cannot overpass 50 RPS, and also they cannot be less than 25 RPS, since the frictional torque is big enough to be exceeded by the engine.

To solve that conflict, the self-controlling function was modified with an internal limiter that took into account these problems. When tested again, the satellite modifies its velocity and adapt it to values that are accepted by the system. However, as suspected, the characteristic time of the process in which all the steps were performed, was exorbitant. The time required by the satellite to perform this process is greater than 10 seconds. Therefore, whenever the satellite reads an angle, computes the required velocity and commands it, it will be located in another position different than the one used, and the commands are no longer useful. This erratic behavior can be seen below in Figure 6.2 where the real position and velocity of the satellite is showed.

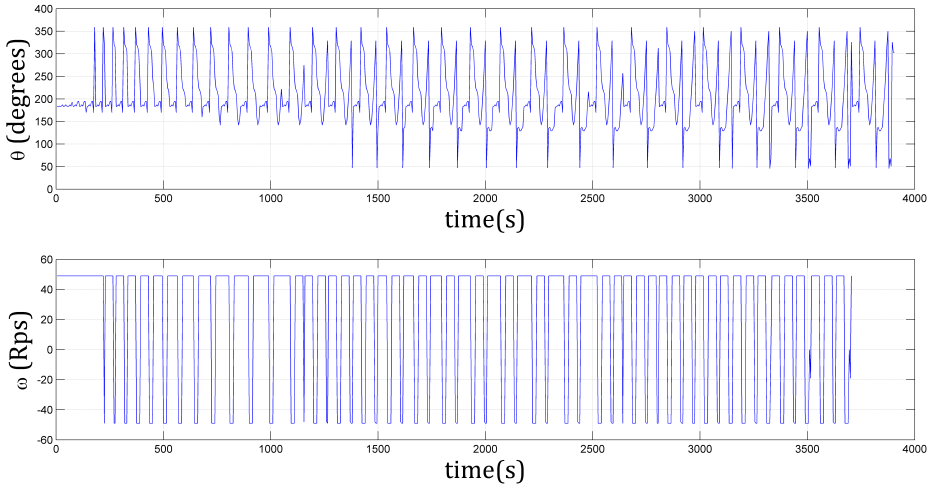


Figure 6.2: Real behavior of the system in Closed-Loop operation.
Information taken from EyasSat's telemetry lines.

Due to this problem, the satellite could never reach the setpoint that is required and therefore it cannot be used with such a PID system. To ensure the hypothesis of the fail was correct, and the real problem of the satellite is its characteristic time, a new Matlab simulator was developed. In that new simulator, the user can set the initial state of the satellite and the desired setpoint, and as before, the own simulator creates ten different cases in which the satellites should know its position, recognizes the shortest path and perform the motion. The only difference now, is that the simulator saves at every instant of time all the state variables of the system. Actually, each step lasts a hundredth part of a second, to ensure it is accurate enough.

The key characteristic of the new simulator is that it allows the user to apply some delay to the system, just by commanding the wheel with the velocity that should be implemented some steps before. Meanwhile, the wheel keeps rotating with the same velocity as it has, simulating that the system is performing the whole cycle and it is waiting for a new command.

The capability of the simulator of modifying the amount of time that the user delays the system allows to know which point the delay is long enough so that the system becomes unstable. First, the system was introduced a delay of one second, but it still converges towards the solution, then the delay was defined to be two seconds, five seconds and ten seconds. During the ten seconds configuration the system could not reach the setpoint whereas for the other ones it could. After some trials, the limit of its stability was defined to be about eight seconds, and any configuration with a delay greater than that will provoke the system to become unstable. It should be remembered that the typical response characteristic time of the system was ten seconds. Below, the reader could find in Figure 6.3 four different cases, showing the stable, the unstable and the limitant conditions. The black line represents the desired attitude of the satellite.

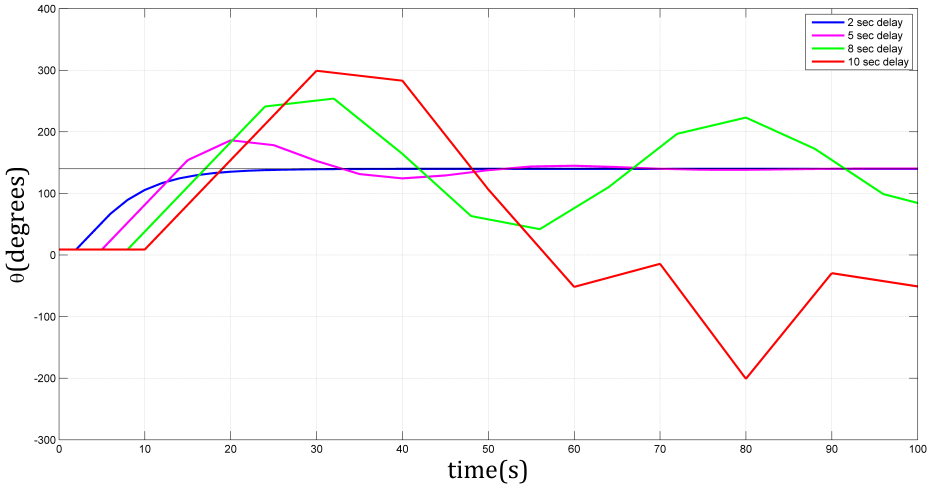


Figure 6.3: Simulations of the delayed system for 2, 5, 8 and 10 seconds.

Chapter 7

Conclusions

7.1 Main results and objectives met

During the performance of that work, the student has fulfilled some objectives such as:

- Obtaining knowledge about real spacecrafts, its subsystems, how they work and the physics that are beneath all their development.
- Improving the theoretical background about Control Engineering.
- Learning different programming languages, how to code different kinds of software such as Matlab, LabView, Latex, etc. and how to join their functionality.
- Development of an EyasSat's GUI that replaces the existing one.
- Creation of different controlling algorithms.
- Development of some testing cases for the algorithms through the GUI of the satellite.
- Elaboration of a formal engineering report and technical documentation.

The main result obtained from that work, seeing the capabilities and response of the system, is that the EyasSat is not an appropriate satellite for its purposes, since it cannot provide a stable response whenever it is required. It is shown to be caused by the internal delay of the system, and it could lead to future works on the EyasSat. Apart from that, the fulfillment of the project includes the development of the Command and Control Center of the EyasSat, answering one of the needs of the system and making easy the approaching of other students to the system.

Besides, there is a lot of possible work that can be done in order to improve the EyasSat either to keep increasing the learning of the student about what is a real space project and how they are developed. This is explained in following section.

7.2 Future work

There are a lot of modifications that could be applied to the final system in order to improve its characteristics and capabilities. As it has been commented during the elaboration of the work there are some fatal errors that influence the performance of the

spacecraft in such a way that it cannot being properly used.

First of all, the stability of the system must be ensured. For that purpose, reducing the characteristic times of the system i.e, if the software spends less time in processing and commanding all the information either the wheel could response fast enough to the modifications dictated by the controller such stability could be achieved. Also, the controller that is embedded should be able to modify and control the torque provided by the engine, so that the system can operate as desired. Finally, a different actuator should be installed on the wheel, i.e, in the ADCS system, so that the reaction wheel would not be so limited as it is.

Whenever a complete control of the satellite is ensured, since the sensors are properly constructed it could enter in operation, therefore a more exhaustive study should be done about the power board and the capability it has for providing energy. There must have knowledge about how the power of the system is handled and check if the power systems are constructed properly.

On the other hand, there could be another line of work, running in parallel with the previous one. Since some requirements of the spacecraft are already known, some engineers could developed a new mission plan, so that the "building and manufacturing" team will have completely defined the required spacecraft. Using all these requirements coming from different sources, a completely new spacecraft can be constructed. Using open source electronic platforms such as Arduino which is easy to learn and specially it is cheap enough for an university project, all the systems could be improved without having to figure out the way of skipping the ones that are already installed on the EyasSat. The electronic components will be cheaper if bought separately and installed by the developing team. Besides, the construction of the system will provide the engineers with some technical background related to mechanical and electrical concepts that are absents in the EyasSat project. Actually, this line of work will be the perfect complement to this project since it allows the engineers to follow all the steps and milestones that are involved in any project. Going from the conceptual design of the whole system until the final construction of it. Finally, if the system is completely defined to enter into operation, the team will have to handle with some authorities so that the system could be certified and authorized. That activities will complete the formation of the team.

7.3 Estimated cost analysis of work done

The estimated costs of the project will include the license of the software that are employed, all the equipments and materials that are required for the performance of the project and the costs of the personal that is involved in its construction. Actually, the personal is formed by an engineer and a laboratory technician.

Total budget of the project	
Product	**Price**
EyasSat hardware	8000 €
Matlab's license	Individual license - 2000 €
Labview's license	Complete license - 3310 €
A tachometer	80 €
Raw material (2 m profiled aluminum bar)	270 €(135€/m)
2 working hours of a laboratory technician	80€(40€/h)
600 working hours of a BSc engineer	25200 €(42€/h)
Total investment of the project	**38940 €**

Table 7.1: Estimated budget of the project.

Appendices

Appendix A

Solar arrays physics

A.1 Physical principles of solar arrays

The working principle of the solar cell arrays is based on the photovoltaic effect. It means the creation of a potential difference at the junction of two different materials due to some radiation. The process can be split in three steps, they are:

1. The absorption of photons in the materials produces some charge carriers.
2. The photo-generated charge carriers are separated at the linkage.
3. Finally, those carriers are joined and collected at the terminals of the joining.

Usually, any solar cell will be formed by an absorber layer in which the incident photons are efficiently absorved, giving birth to the electron-hole pair. In order to separate the electrons and the holes from each other, a semipermeable membrane should be installed at both sides of the absorber. This component will allow to pass through it one kind of charger, separating them perfectly. In order to make possible this separation, the pairs should reach the membrane. It happens because the thickness of the absorber layer is smaller than the diffusion length of the charge carrier.

If the membrane is selected to block the holes and let electrons pass, it has to have a large conductivity for the later ones. This property is a characteristic of the n-type semiconductor, due to the large difference in electron and hole concentrations. Since the conductivity of holes is not favored by the material, they should be moved due to the recombination processes. The opposite happens if a p-type semiconductor is selected as a hole membrane.

For reducing the injection of holes from the absorber into the n-type semiconductor, an energy barrier should be introduced in the valence band, called ΔE_v .

The valence band is the highest range of electron energies in which electrons are normally present at absolute zero temperature. The barrier should be located between the n-type semiconductor and the absorber. Ideally, this can be achieved by choosing an n-type semiconductor that has a larger band gap than the absorber's one and therefore the energy difference is created in the valence band of both materials.

For suppressing the injection of electrons into the p-type semiconductor, the band gap of the absorber is required to be smaller than the one of the p-type semiconductor. Furthermore, the p-type should have the band off-set in the conduction band,

called ΔE_c. The conduction band is the lowest range of vacant electronic states. It should be located also between the absorber and the p-type semiconductor. For having the band off-set in the conduction band the electron affinity, χ_e, of the p-type semiconductor should be smaller than the electron affinity of the absorber. For a semiconductor-vacuum interface (that is, the surface of a semiconductor), electron affinity is defined as the energy obtained by moving an electron from the vacuum just outside the semiconductor to the bottom of the conduction band just inside the semiconductor. The additional advantage applying membrane materials with large band gaps is letting almost all photons be transmitted and absorbed into the absorber.

The asymmetry in the electronic structure of the n-type and p-type semiconductors is the basic requirement for the photovoltaic energy conversion. The terminals or, in other words, electrodes of the solar cell are attached to the membranes. We refer to the structure between the terminals as a junction and the above described solar cell structure is denoted as a single junction solar cell. The quasi-Fermi level for electrons, called EFC, and the quasi-Fermi level for holes, the EFV, are used to describe the illuminated state of the solar cell. The energy difference between the quasi-Fermi levels is a measure of efficient conversion of energy of radiation into electrochemical energy.

The solar cell array can work in two different configurations. The first is the open-circuit condition, when the terminals of the solar cell are not connected to each other and therefore no electric current can flow through an external circuit. At this condition, a voltage difference can be measured between the terminals of the solar cell. This voltage is denoted the open-circuit voltage, V_{oc}, and it is an important parameter that characterizes the performance of solar cells. The other condition is the short-circuit one. In this case the terminals of the solar cell are short circuited and a current flows through the external circuit. This current is denoted as the short-circuit current, I_{sc}. The I_{sc} is also an important parameter that characterizes the performance of solar cells.

- **The p-n junction**

The most frequent example of the described solar cell structure is produced with crystalline silicon (c-Si). A moderately doped p-type c-Si is used as the absorber. On the top side of the absorber a thin, highly-doped n-type layer is formed as the electron membrane. On the back side of the absorber a highly-doped p-type serves as the hole membrane. At the interfaces between the c-Si p-type absorber and the highly-doped n-type and p-type membranes, some regions are formed with an internal electric field. These regions are especially important for solar cells and are known as p-n junctions. The presence of the internal electric field in the solar cell facilitates the separation of the photogenerated electron-hole pairs. When the charge carriers are not separated from each other in a relatively short period of time they will be annihilated in a process that is called recombination and thus, they will not contribute to the energy conversion. The easiest way to separate charge carriers is to place them in an electric field. In the electric field the carriers having opposite charge are drifted from each other in opposite directions and can reach the electrodes of the solar cell. The electrodes are the metal contacts that are attached to the membranes.
The p-n junction fabricated in the same semiconductor material such as c-Si is an example of the p-n homojunction. There are also other types of a junction that result in the formation of the internal electric field in the junction. The p-n junction that is formed by two chemically different semiconductors is called the p-n heterojunction. In the p-i-n junctions, the region of the internal electric field is extended by inserting an intrinsic, i, layer between the p-type and the n-type layers. The i-layer behaves like

a capacitor and it stretches the electric field formed by the p-n junction across itself. Another type of the junction is a junction between a metal and a semiconductor, MS junction.

- **Formation of a space-charge region in the p-n junction.**

In the n-type pieces, either in the p-type ones, the charge neutrality is maintained. In the n-type semiconductors, the large concentration of negatively charged free electrons is compensated by positively-charged ionized donor atoms. In the p-type semiconductor holes are the majority carriers and the positive charge of holes is compensated by negatively-charged ionized acceptor atoms.

For the isolated n-type semiconductor we can write:

$$n = n_{n0} \approx N_D \tag{A.1}$$

$$p = p_{n0} \approx n_i^2 / N_D \tag{A.2}$$

Hence, for the isolated p-type semiconductor:

$$p = p_{p0} \approx N_A \tag{A.3}$$

$$n = n_{p0} \approx n_i^2 / N_A \tag{A.4}$$

When a p-type and an n-type semiconductor are brought together, a very large difference in electron concentration between both semiconductors causes a diffusion current of electrons, from the n-type material, across the metallurgical junction, into the p-type material. Similarly, the difference in hole concentration causes a diffusion current of holes from the p-type to the n-type material. Due to this diffusion process the region close to the metallurgical junction becomes almost completely depleted of mobile charge carriers. The gradual depletion of the charge carriers gives rise to a space charge created by the charge of the ionized donor and acceptor atoms that is not compensated by the mobile charges any more. This region of the space charge is called the space-charge region or depleted region. Regions outside the depletion region, in which the charge neutrality is conserved, are denoted as the quasi-neutral regions.

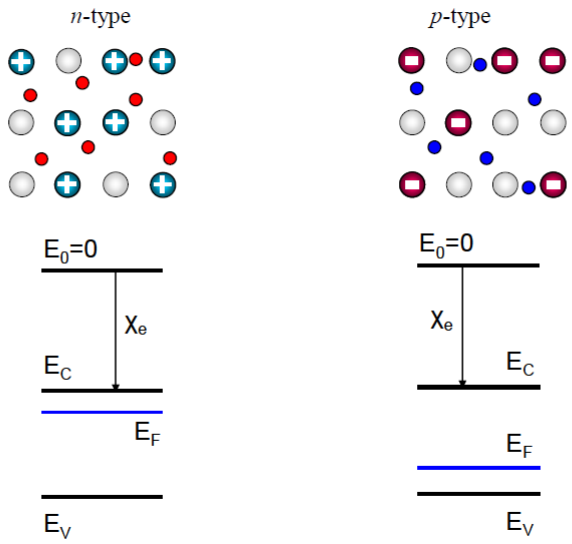


Figure A.1: Isolated n-type & p-type semiconductors and their corresponding valence bands.

The space charge around the metallurgical junction results in the formation of an internal electric field which forces the charge carriers to move in the opposite direction than the concentration gradient. The diffusion currents continue to flow until the forces acting on the charge carriers, namely the concentration gradient and the internal electrical field, compensate each other. The driving force for the charge transport does not exist any more and no net current flows through the p-n junction.

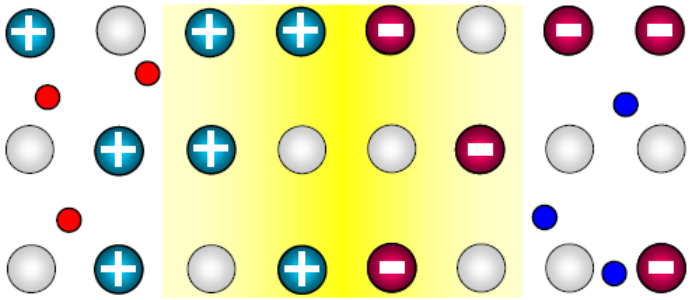


Figure A.2: Junction formed by a p-type and an n-type semiconductor. The colored part is a space-charge region.

A.2 Solar arrays in EyasSat

According to the configuration of the solar array it can be modeled as a circuit formed by a generator, a diode and some resistances. That circuit can be seen in the Figure A.3.

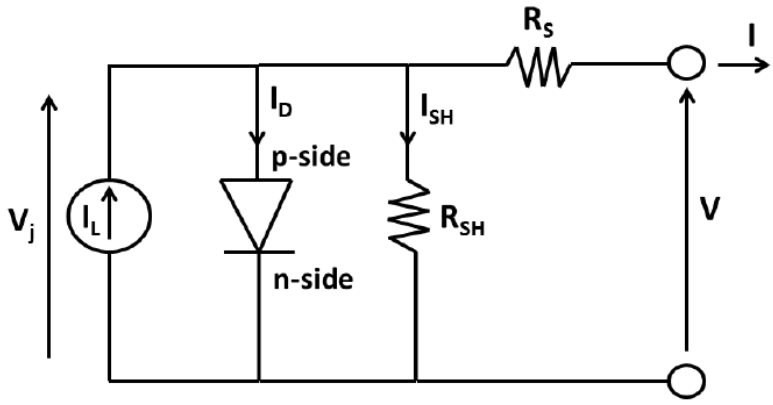


Figure A.3: Equivalent circuit of the solar panel

The modelization is composed by some elements:

- A current generator whose intensity is called I_L. Such intensity will depend on the size of the cell but also on the type of radiation spectrum being absorbed. A photon, depending on its energy, is capable of generating only one electron/hole pair at most and only if its frequency and corresponding energy is high enough to knock out an electron from the valence into the conduction band. Any excess energy of the photon is converted into thermal energy, which has a negative impact on the solar array (the energy conversion efficiency lowers when the array temperature increases). This consequence will be explained later on.

- A diode with saturation current I_0. The p-side of the junction represents the positive end of the diode (just like for the model of Figure **??**).

- A series resistance R_S modeling the effect of the front and back metallic contacts of the solar cell.

- A shunt resistance R_{SH} modeling other types of losses not considered herein.

The total current flowing out of the solar panel can be expressed as the illumination current minus the current flowing through the diode model and through the shunt resistance.

The circuit can be defined physically by its current-voltage characteristic (I-V) shown in Figure A.4. This curve allows the reader to check the behavior and therefore the response of the circuit whenever some stimulation affects it.

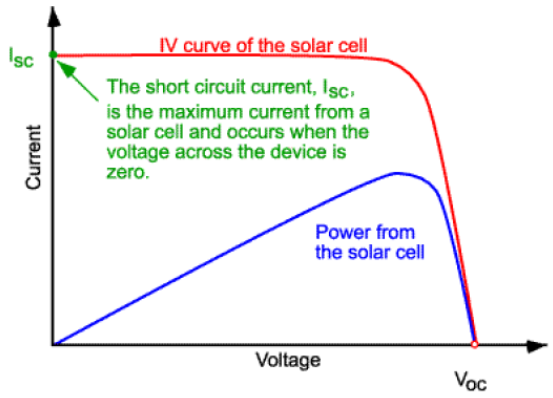


Figure A.4: I-V diagram of a solar array

Since the current flowing through a diode is also a function of temperature following the relationship defined as:

$$\begin{cases} I_D = I_0[exp(\frac{qV_D}{nkT}) - 1] & \text{for} \quad V_D > 0 \\ I_D = -I_0 & \text{for} \quad -V_{BD} < V_D < 0 \end{cases} \tag{A.5}$$

Where V_D is the applied voltage to the diode, V_{BD} is the breakdown voltage, k is the Boltzmann constant, T the junction temperature and n is the ideality factor (we shall assume that it is 1 for simplicity).

Therefore, the temperature at which the solar cell array is going to work should be considered. A characterization of this phenomena can be seen on Figure A.5.

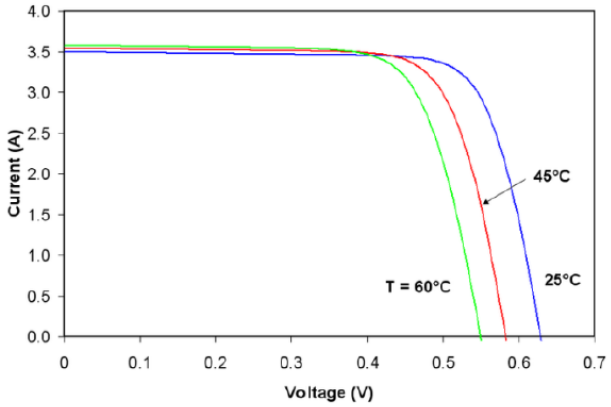


Figure A.5: Temperature effects on I-V diagram of the solar cell

Bibliography

[1] P. Fortescue, G. Swinerd, and J. Stark, *Spacecraft systems engineering*. John Wiley & Sons, 2011.

[2] G. M. Burditt, "Communications with manufacturing company of eyassat." Sales & Technical Support.

[3] F. Cichocki, "Solar cell. operational principles," tech. rep., 2014.

[4] F. Cichocki, "Eyassat. Memory guides," tech. rep., 2014.

[5] W. J. Larson and J. R. Wertz, "Space mission analysis and design," tech. rep., Microcosm, Inc., Torrance, CA (US), 1992.

[6] J. R. Wertz, D. F. Everett, and J. J. Puschell, *Space mission engineering: the new SMAD*. Microcosm Press, 2011.

[7] K. Ogata and Y. Yang, "Modern control engineering," 1970.